UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

GENERADOR DE TERRENOS CREADOS A PARTIR DE ELEMENTOS GEOGRÁFICOS REALES

Asier Mahave Estévez
Tutor: Carlos Aguirre Maeso

JUNIO 2019

GENERADOR DE TERRENOS CREADOS A PARTIR DE ELEMENTOS GEOGRÁFICOS REALES

AUTOR: Asier Mahave Estévez
TUTOR: Carlos Aguirre Maeso

Dpto. Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Junio de 2019

Resumen

Este Trabajo Fin de Grado consiste en la realización de un programa que, tomando como referencia dos puntos de coordenadas del globo, genere un mapa de altura interpretable por el motor de videojuegos **Unity**, de tal manera que se pueda modelar un terreno 3D en base al mundo real. Para ello se toman las alturas de diversos puntos comprendidos por el rectángulo con esquinas superior izquierda e inferior derecha definidas por las coordenadas, se procesan dichas alturas y, dependiendo de su valor, se les asigna un color en la escala de grises. Una vez generada la imagen, tan solo queda convertirla al formato *raw*, condición exigida por Unity, e importarla al proyecto deseado.

Éste trabajo nace inspirado por la propensión de los medios digitales como los videojuegos a tornarse cada vez más realistas, tomando como ejemplo el mapa del título Grand Theft Auto V, videojuego situado en una reproducción bastante impresionante de la ciudad estadounidense de Los Ángeles. Gracias a esta herramienta se puede generar de manera cómoda y rápida un mapa de altura de la zona deseada para obtener mediante software de modelado 3D, terrenos con los que agilizar el desarrollo de un videojuego o bien usar como base de mapas interactivos aprovechando el hecho de que son mapas de áreas geográficas reales, abaratando los costes de este tipo de proyectos, haciéndolos así más accesibles a equipos más pequeños, que es la tendencia actual, ya que la opción de desarrollar manualmente los mapas es demasiado costosa y probablemente menos precisa.

Palabras clave

Formato raw, formato png, API, Unity, Mapa de altura, Modelado 3D.

Abstract

This Bachelor Thesis purpose is to create a software able to generate a heightmap from two real word coordinates given, which can be translated by the videogame engine **Unity** into a 3D terrain modeled after the square defined by the specified coordinates of the real world. To do this it processes all the heights from all the points inside the rectangle defined by two coordinates, one for the upper left corner the other for the bottom right corner. Once all the heights are structured and normalized they are assigned a grayscale color depending on their value. After this we can generate a .png format image, so the only thing left is convert it to .raw format in order for Unity to be able to read it.

This project has been inspired by the inclination of the digital media like, for example, videogames, to become more and more realistic. Taking a look at the map of Grand Theft Auto V, a videogame that takes place in a very accurate representation of Los Angeles we can see a clear example of what I mean. Thanks to this tool, a heightmap can be generated in a quick and easy way that we can use to generate a 3D terrain model which can speed up the development of a videogame or it can be used to create a interactive map taking advance of it being made after the real world data, lowering the cost for this kind of projects, making them more accessible for small teams, as it is much more accessible than manually modeling every bit of the map, which is more expensive and probably less precise.

Keywords

Raw format, Png format, API, Unity, Heightmap, 3D Modeling.

INDICE DE CONTENIDOS

INDICE DE FIGURAS

1 Introducción

1.1 Motivación

Desde sus modestos orígenes con títulos como “Pong” los videojuegos han ido evolucionando y ganando poco a poco popularidad hasta convertirse en una industria en todas condiciones. Hoy en día se puede nombrar multitud de empresas multinacionales como Nintendo o Sony, establecidas ya en el comercio, que publican anualmente multitud de videojuegos. Estos, como todos, han ido evolucionando a pasos agigantados, desde el ya mencionado “Pong” hasta “Grand Theft Auto V”, haciéndose cada vez más y más complejos, creciendo los equipos para cubrir los distintos elementos necesarios hoy en día, entre los que se incluyen programadores, diseñadores, guionistas, músicos, …

Con esta popularidad, y la mayor accesibilidad que se tiene cada año a herramientas profesionales de desarrollo, cada vez es más notorio el mercado de los juegos *Indie*, concepto que hace referencia a equipos independientes de empresas grandes, que aprovechan la tecnología que tienen a su alcance (por ejemplo, el motor de videojuegos Unity, desarrollado por Unity Technologies) para producir juegos de menor calibre, con la originalidad como característica destacable. Ejemplos de estos equipos son el estudio Wube Software, con **Factorio** o **Slay the Spire** de Mega Crit, ambos títulos con más de millón y medio de copias vendidas, Edmund McMiller, Florian Himsl y Nicalis con **The Binding of Isaac**, con más de dos millones de copias vendidas, así como **Cuphead** de la mano del estudio StudioMDHR con más de 3 millones.

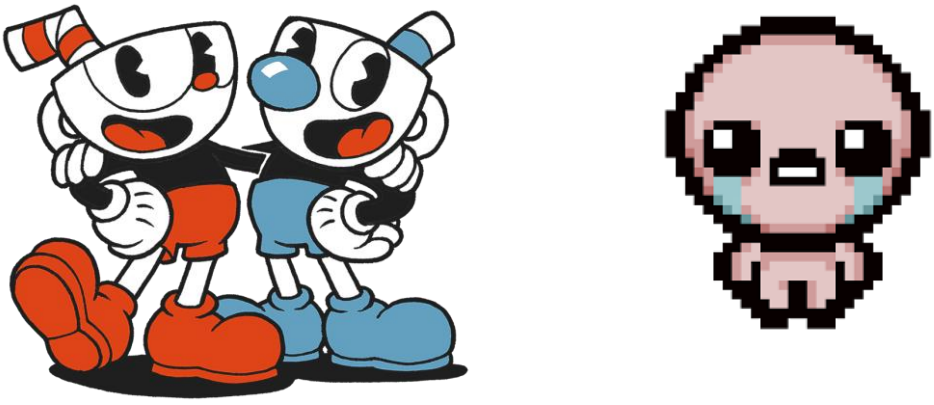

Figura 1-1: (Izquierda) Protagonistas Cuphead recuperado de http://www.cupheadgame.com/img/cuphead_header.png. (Derecha) Protagonista The Binding of Isaac recuperado de https://vsbattles.fandom.com/wiki/Isaac_(Binding_of_Isaac)

Cada vez aumenta más el alcance de este tipo de desarrollo independiente por el mismo presupuesto, pero la barrera de los videojuegos indie sigue siendo el desarrollo en un mundo más realista como el ya mencionado GTA V. Ése es el motivo por el que me he

decidido a desarrollar un prototipo de herramienta que permita reproducir el terreno del área escogida del planeta de una manera rápida y sencilla.

En esta memoria de TFG se detallará el proceso seguido para el desarrollo de la herramienta de generación de mapas de altura. Así mismo se explicarán los conceptos en los que se basa y los distintos enfoques de realización, herramientas y tecnologías que he considerado a lo largo de su producción.

1.2 Objetivos

El objetivo final de este TFG es la creación de una herramienta que permita generar mapas de altura de áreas geográficas reales en un formato admitido por el motor de videojuegos Unity, a fin de que éste genere un modelo 3D de dicha localización listo para ser editado a gusto del usuario e integrado en su proyecto.

El objetivo final de estos mapas para este estudio en concreto es su integración en proyectos de Unity, pero las posibilidades son tantas como quiera darle el usuario, ya que los mapas de altura presentan más utilidades como la posibilidad de importarlos en videojuegos como el simulador de construcción *Cities: Skylines* o en multitud de programas de modelado 3D como Blender.

Figura 1-2: Logotipo Blender recuperado de https://www.blender.org/

1.3 Organización de la memoria

La memoria a partir de este punto consta de los siguientes capítulos:

- **Estado del arte**

Apartado en el que explico los conceptos que conciernen este trabajo de fin de grado como son las tecnologías de mapas interactivos de Leaflet o el motor de videojuegos Unity y entro en detalle sobre qué son los mapas de altura.

- **Diseño**

Comienza con un breve repaso por las tecnologías que he considerado y acabado usando a lo largo del desarrollo del trabajo para acabar con el uso de estas tecnologías en la organización y función de los distintos módulos en los que he dividido la herramienta.

- **Desarrollo**

Explicaciones más detalladas y técnicas sobre la lógica tras cada módulo que conforma el proyecto.

- **Integración, pruebas y resultados**

Contiene la explicación de la interacción entre los módulos detallados previamente, así como las pruebas realizadas para comprobar la corrección del proyecto y los resultados de las mismas.

- **Conclusiones y trabajo futuro**

Para finalizar comento lo aprendido en el transcurso del trabajo y los cabos que atar en un futuro para seguir mejorando la herramienta desarrollada.

2 Estado del arte

Las tecnologías y conceptos involucrados en el desarrollo y funcionamiento del proyecto podrían dividirse en dos tipos: mapas interactivos, con los que he desarrollado la toma de coordenadas, y mapas de altura y modelos tridimensionales, que conforman el producto final. Resumo ambos a continuación.

2.1 Mapas Digitales

Los mapas digitales son, como su nombre indica, representaciones de un área con posibilidad de albergar distintos tipos de información, por ejemplo, geográfica, política, de distintos parámetros como el índice NDVI utilizado con frecuencia en la agricultura, etc. Sus usos son muchos y muy variados. En la actualidad hay distintas herramientas que te permiten desarrollar y personalizar tu propio mapa digital, yo me he decantado para la toma de coordenadas por desarrollar con Leaflet un mapa digital interactivo que detallaré más adelante.

2.1 Leaflet

Leaflet es una librería de JavaScript gratuita y open-source publicada el 13 de mayo de 2011 centrada en la aportación de mapas interactivos, desarrollada originalmente por Vladimir Agafonkin, apoyada más tarde por una comunidad abierta de contribuyentes con el objetivo principal de aportar una alternativa a la API de Google Maps, que como ocurre con la Google Elevation API, es de pago.

Figura 2-1: Logotipo Leaflet recuperado de https://leafletjs.com/

2.2 Mapas Interactivos

Los mapas interactivos son modelos digitales de áreas geográficas que permiten al usuario navegar por los mismos desplegando distintas opciones informativas. Un ejemplo claro de esto es Google Maps o, por ejemplo, el plano de la terminal T4 del aeropuerto de Madrid-Barajas, en el que puedes cambiar la vista a las distintas plantas, así como visualizar los puntos de interés como comercios o zonas de embarque haciendo clic en los distintos iconos.

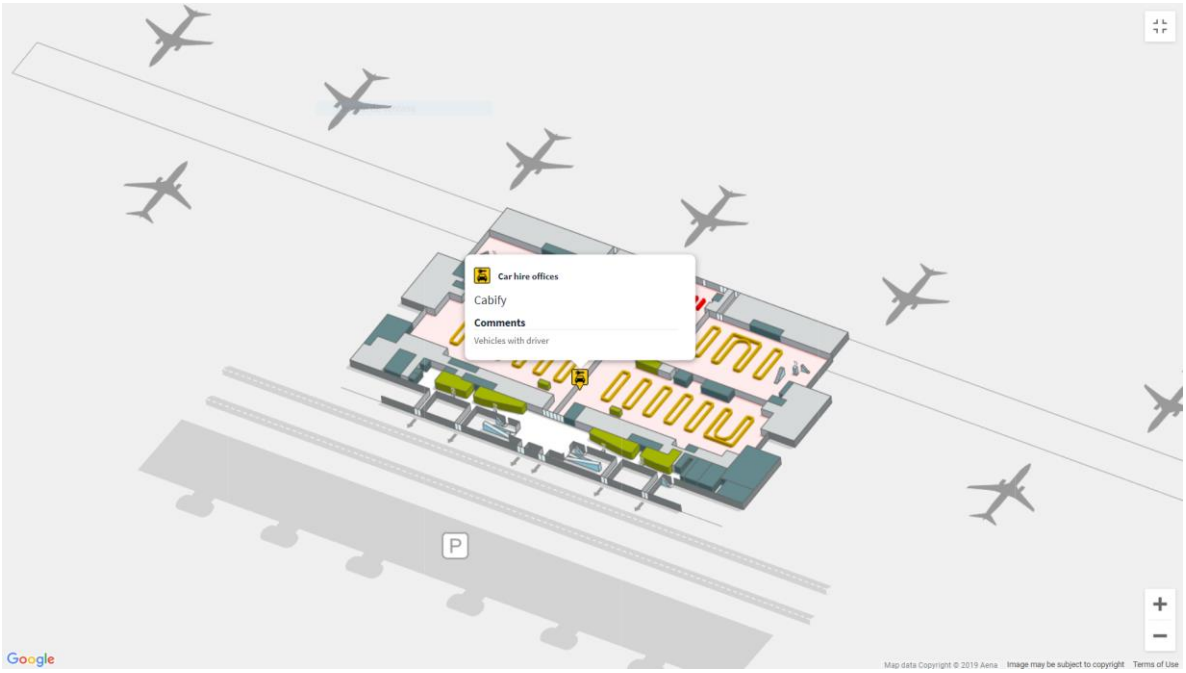

Figura 2-2: Mapa interactivo terminal T4 aeropuerto Madrid-Barajas recuperado de http://www.aena.es/csee/Satellite?Language=EN_GB&ca=MAD&pagename=cartografia&ps=t

2.3 Modelado 3D

"El modelado 3D es el proceso de desarrollo de una representación matemática de cualquier objeto tridimensional (ya sea inanimado o vivo) a través de un software especializado. Al producto se le llama modelo 3D." [6] Este tipo de tecnología, como vemos, permite generar dichos modelos tridimensionales a partir de múltiples puntos conectados entre sí. Esta tecnología encuentra su aplicación en múltiples disciplinas, como la medicina donde representar modelos 3D de órganos o prótesis, la arquitectura para prototipar construcciones, la ingeniería industrial para modelar diseños, el cine a la par que el mundo del videojuego para crear animaciones y un sinfín más de posibilidades.

2.4 Unity

Unity es un motor de videojuegos multiplataforma para MAC y Windows lanzado al mercado en el año 2005 con la idea de hacer más accesible el desarrollo de videojuegos. A lo largo de los años Unity ha ido ganando popularidad por la enorme utilidad que aporta a los usuarios a la hora de crear videojuegos tanto en 2D como en 3D con C# y JavaScript como lenguajes de programación.

Unity contiene una multitud de herramientas de modelado de assets o ítems, permitiendo por ejemplo el coloreado de los diferentes objetos que conforman el proyecto con distintas texturas, aplicación de animaciones a los mismos, así como modelado de terrenos y soporte para su generación mediante mapas de altura. Una de las características más destacables de este programa es la existencia de una tienda integrada, la **Asset Store**, cuyo contenido abarca todo tipo de productos, desde texturas a herramientas más sofisticadas, normalmente de pago.

Figura 2-3: Logotipo Unity recuperado de https://unity.com/

2.5 ¿Qué es exactamente un mapa de altura?

Un mapa de altura es una representación de la elevación de un terreno con una escala de colores. Los más habituales, y los protagonistas de este proyecto son los basados en una escala de grises. El número de colores que conforman la escala es 256, siendo así por estar limitada a los 8 bits del formato RGB (8 bits para cada banda, un total de 256 valores distintos), ya que para conseguir una escala de grises los valores de las bandas de color rojo, verde y azul han de tener el mismo valor, dando lugar a esa cifra.

Cada color de la escala pues representa una altura. En la escala de grises para los mapas de altura, el blanco es el punto más alto y el negro el más bajo, el resto de los puntos se distribuirán de manera relativa a dicho rango. Esta representación puede dar lugar a cierta imprecisión, ya que se pierde, por ejemplo, la altura total del terreno, al tomar como la altura cero el punto más bajo. También puede dar lugar a representaciones más escalonadas que den lugar a una representación más abrupta.

El uso principal de los mapas de altura, como se ha comentado en la introducción, es la representación de la elevación de un terreno de una manera sencillamente

interpretable por programas de modelado 3D para que estos puedan generar un modelo tridimensional de dicho terreno.

2.6 ¿Por qué se usan?

Si bien la limitación de colores puede provocar los problemas comentados más arriba, los mapas de altura siguen siendo una manera muy ligera, rápida de cargar y accesible (pueden generarse o modificarse con programas de edición de imágenes) a la hora de representar terrenos, frente a otras alternativas como la interpretación de paquetes XML o formatos del estilo, que resultan más pesados y complejos de interpretar por un ser humano.

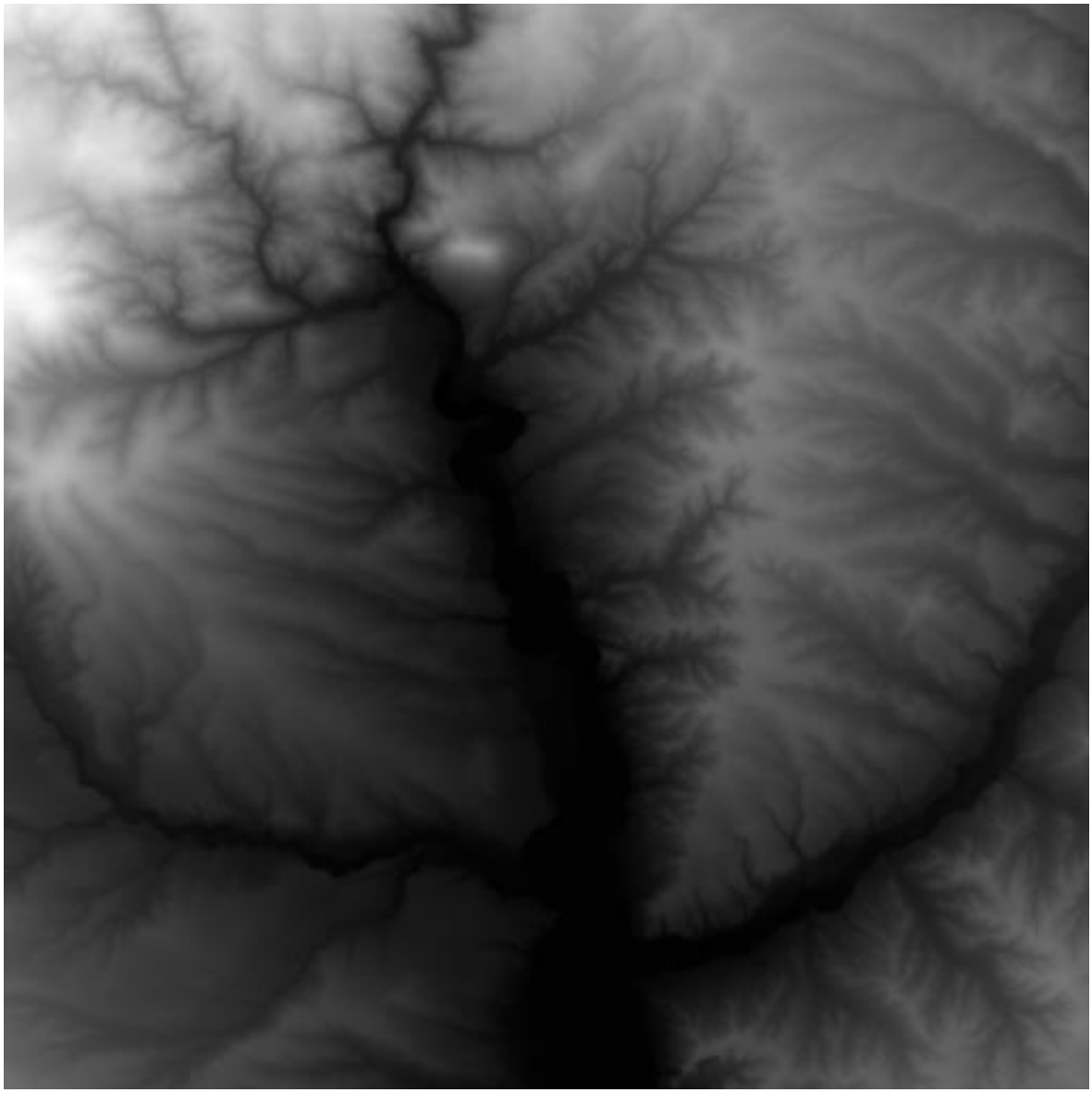

Figura 2-4: Mapa de altura de El Pardo

3 Diseño

El diseño de esta herramienta busca generar una imagen en formato. raw, a partir de dos coordenadas, para ello son necesarios distintos componentes:

- El lenguaje de programación.
- Una fuente de la que obtener las alturas y coordenadas.
- Un módulo de creación y conversión de imágenes.

3.1 Tecnologías a usar

El lenguaje de programación empleado, debido a su versatilidad, es Python, más concretamente la versión 3.7.

Para la obtención de las alturas, la fuente finalmente escogida ha sido Google Elevation API, una API para Python de Google que proporciona los datos de elevación del punto geográfico solicitado. Si bien cumple de sobra las expectativas, el uso gratuito de esta fuente supone ciertas limitaciones respecto al número de solicitudes que se pueden realizar, dando lugar a la formación de una imagen al día de una resolución máxima de 512 píxeles de alto y de ancho.

Figura 3-1: Tecnologías usadas

Para la creación de la imagen a partir de las alturas, la librería de Python TkInter (Tk Interface) se ajusta a nuestras necesidades, permitiendo definir cada píxel de la imagen a crear, en un principio en formato .png. Para la conversión desde el formato .png a .raw, el programa utilizado es ImageMagick, un software "open source" de tratamiento de imágenes que soporta, entre otros, dicha conversión.

3.2 Tecnologías desechadas

A lo largo del proceso de investigación, el objetivo del trabajo estaba claro, pero la manera de afrontarlo estaba completamente abierta. Los distintos caminos considerados para conseguir generar terrenos a partir de elementos geográficos reales son los siguientes:

- Importar paquetes de datos XML desde Open Street Map, un servicio de mapas de acceso libre.
- Importar mapas de altura ya creados desde el entorno Earth Engine de Google, en concreto de la base de datos de elevación digital Shuttle Radar Topography Mission (SRTM), la cual aporta un mapa de altura global, del que tan solo habría que extraer las partes requeridas.
- Generar nuestro propio mapa de altura, píxel a píxel, con los datos de alturas recabados de una fuente fiable.

La primera decisión fue descartada por el enfoque distinto al proyecto original, aparte de limitaciones de versatilidad con respecto a las otras dos, ya que se prescinde del formato ya hablado de los mapas de altura para dar paso a la lectura de paquetes en formato XML más pesada y costosa.

La segunda decisión aportaba baja resolución en las imágenes finales, ya que no parece estar pensado para recreación, por ejemplo, de una región de varios kilómetros cuadrados.

La última, la finalmente escogida, presentaba el problema de escoger una buena fuente para obtener los datos de altura, ya que son un recurso bastante costoso. Fuentes como Leaflet, librería “open source” de JavaScript, no recaban datos de altura.

Figura 3-2: Tecnologías desechadas

3.3 Modularización

El proyecto se encuentra estructurado en cuatro módulos, uno para cada fase de la ejecución, empezando por la interfaz gráfica en la que se permite al usuario escoger cómo introducir las coordenadas, el siguiente centrado en la obtención de alturas en base a las coordenadas requeridas. Tras la toma de datos otro módulo los interpreta estructurándolos en una matriz y, finalmente el último, transforma las alturas en colores con los que genera la imagen definitiva.

3.3.1 Módulo de interfaz gráfica

3.3.1.1 Introducción de coordenadas

Este módulo cumple el propósito de permitir al usuario la introducción de las dos coordenadas geográficas en formato latitud, longitud en grados decimales. Para ello despliega una primera ventana en la que se muestran dos botones, uno para introducir las coordenadas por medio de un archivo de texto, otro para introducir directamente en un formulario las coordenadas.

El formato del archivo de texto para importar las coordenadas sigue el siguiente formato: Latitud coordenada superior izquierda, longitud coordenada superior izquierda, latitud coordenada inferior derecha, longitud coordenada inferior derecha

Este archivo se puede construir a mano o haciendo uso del mapa digital que he creado para dicho propósito haciendo uso de la librería open-source **Leaflet** para mostrar un mapa interactivo sobre el cual se puede seleccionar un área haciendo clic sobre los puntos de interés, descargando las coordenadas de dicha zona con el nombre de archivo escogido.

La idea detrás de la necesidad de dos puntos geográficos consiste en poder formar un área rectangular, tomando como lado superior e inferior del mismo la diferencia entre ambas longitudes y de altura esta vez la diferencia entre las latitudes. En el caso de hacer uso del mapa interactivo para generar las coordenadas, como se puede apreciar si se seleccionan dos puntos, la aplicación generará un cuadrado perfecto. Esto resulta útil ya que, como ya veremos, el servicio de obtención de alturas tiene ciertas limitaciones que restringen la toma de alturas por día, lo cual se resume en la creación de una imagen cuadrada de 512 píxeles de alto y de ancho diaria, por lo que en el caso de que el área escogida sea más rectangular se apreciará una deformación en la imagen final.

Figura 3-3: Latitud y longitud en el globo, recuperado de https://keydifferences.com/difference-between-latitude-and-longitude.html

3.3.1.2 Mapa interactivo para la selección de coordenadas

El mapa interactivo sigue un diseño simple, en el que se muestra el mapa base ofrecido por Leaflet que incluye los nombres de los distintos puntos de interés del mapa como países, carreteras o accidentes geográficos.

En este mapa he desarrollado un sistema de marcadores para seleccionar los dos puntos cardinales, de tal manera que al hacer clic la primera vez sobre el mapa se muestra un puntero con un texto que indica la necesidad de seleccionar un último punto para completar el área. Con este sistema interactúan tres botones:

- **Corregir Coordenadas**, disponible tras definir los dos marcadores, traza y muestra el área final escogida corregida para ser un cuadrado perfecto.

- **Limpiar Coordenadas**, disponible siempre que haya al menos un marcador, su funcionalidad es la de eliminar los marcadores escogidos en caso de error o cambio de idea.

- **Descargar Coordenadas**, disponible tras definir los dos marcadores, crea un archivo de texto con el formato adecuado y las coordenadas indicadas bajo el nombre que el usuario le indique en un campo de texto que se despliega con dicho fin.

Estos botones junto con los marcadores proveen al usuario de la funcionalidad básica para obtener el archivo mencionado de una manera cómoda y sencilla.

3.3.2 Módulo de obtención de alturas

Una vez obtenidos los dos puntos geográficos dados por el usuario, este módulo divide el área calculada en 512 bandas a lo ancho, y cada una de esas bandas a su vez en 512 puntos, esta medida si bien parece arbitraria se basa en las limitaciones de la fuente de obtención de alturas, **Google Maps Elevation API**, la cual permite por consulta hasta un máximo de 512 puntos distintos.

El procedimiento pues, una vez separado en dichas bandas el terreno, consiste en la realización de 512 peticiones que cubran por columnas de altura igual a la diferencia de latitudes (la mayor menos la menor), y de izquierda a derecha.

Cada una de las peticiones me devuelve una respuesta en formato JSON con las alturas de 512 puntos con el siguiente estilo:

{"results": [{"elevation": 882.2158203125, "location": {"lat": 40.626982, "lng": -3.847961}, "resolution": 19.08790397644043}, …

Una vez obtenidas todas las respuestas JSON, estas se escriben en un fichero con el mismo formato con el que se han recibido.

Tras ello se despliega un último formulario con el que se le solicita al usuario el nombre con el que va a querer guardar la imagen generada.

3.3.3 Módulo de estructuración de alturas

Tomando el fichero con las respuestas en formato JSON, este módulo extrae cada una de las alturas y las introduce en una matriz. Una vez generada dicha matriz comienza la normalización de los datos para que estos queden en una escala de valores entre 0 y 255.

De la mano de la generación de la matriz van dos funciones para convertir dicha escala de valores en colores en formato hexadecimal, siendo, por ejemplo, 0x00 para los espectros R, G y B, el color 0x000000, es decir, el color negro y 0xFF el color blanco o 0xFFFFFF.

3.3.4 Módulo de generación y conversión de imagen

Este módulo es el encargado de, tomando la matriz de alturas normalizadas, ir, píxel por píxel, transformando la altura al color y plasmándola en el lienzo aportado por la librería **TkInter** de Python.

Una vez generada la imagen, esta se guarda en formato png con el nombre escogido por el usuario en la carpeta “output” del proyecto. Tras ello, se guarda también en el mismo directorio la imagen en formato raw, convertida mediante un comando del programa de edición de imagen gratuito **ImageMagick.**

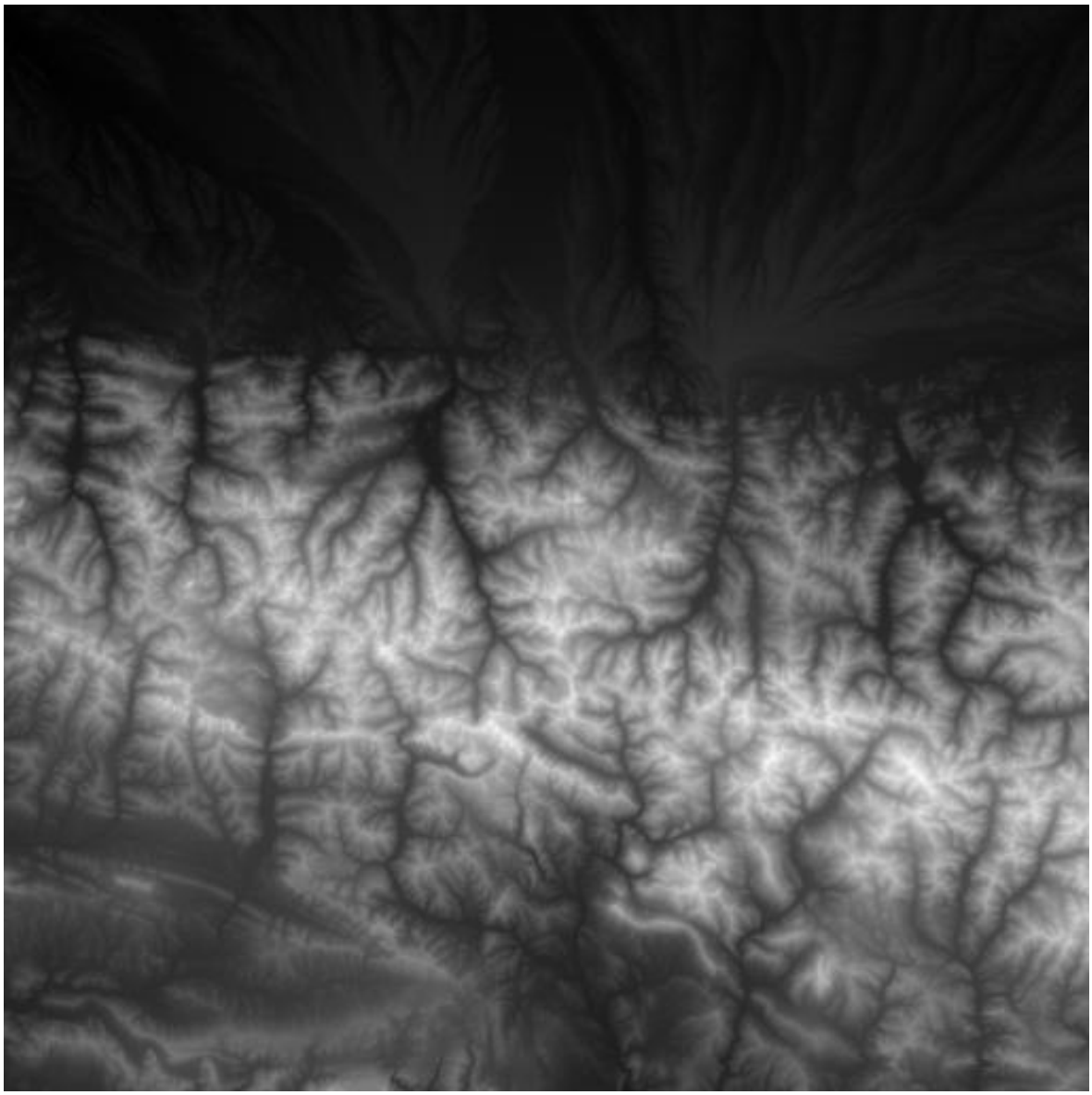

Figura 3-4: Mapa de altura de los Pirineos generado

4 Desarrollo

El desarrollo del proyecto empezó con el módulo de obtención de alturas, en el que tuve que adaptarme al uso de la API antes de dar el siguiente paso, que fue el de estructuración de las alturas. Tras este módulo tenía ya todo lo necesario para generar la imagen, por lo que desarrollé el módulo encargado de dicha tarea. Lo último que desarrollé fue la interfaz gráfica, ya que me centré en la facilidad de uso tras quedar satisfecho con la funcionalidad.

4.1 Módulo de interfaz gráfica

Como hemos comentado, este módulo se centra en facilitar la interacción del usuario con la aplicación. Para ello he hecho uso de la librería de Python **easygui** que nos aporta una serie de herramientas de interfaz, con la cual genero los menús que permiten al usuario escoger el modo de introducir las coordenadas deseadas, así como introducirlas como tal.

Para el primero **easygui** nos aporta la clase buttonbox, la cual muestra la pregunta de "¿Cómo desea introducir las coordenadas?", así como las dos posibles respuestas: "Importar fichero" o "Escribir a mano". La primera de estas opciones supone abrir el explorador de archivos del ordenador en el que se ejecuta para escoger el fichero, para ello **easygui** nos aporta la función *fileopenbox*, la cual nos devuelve la dirección del archivo que el usuario ha escogido, con lo cual podremos partir de eso para leer el archivo, y, por lo tanto, obtener las coordenadas.
La segunda de las opciones despliega un formulario creado mediante la clase *multenterbox* de **easygui**. Este formulario constará de cuatro campos a rellenar, así como cuatro etiquetas que indican el dato a introducir, en nuestro caso las coordenadas de latitud superior izquierda, longitud superior izquierda, latitud inferior derecha y longitud inferior derecha. Una vez relleno, si se envía el formulario, los datos introducidos se pueden recoger en forma de lista con longitud cuatro, de tal manera que habremos conseguido también las coordenadas de la región de interés.

En ambas opciones, las coordenadas recibidas se obtienen como *strings*, por lo que es necesaria su conversión con la función de Python *float()*. También se realiza una comprobación de la corrección de los datos, siendo esto comparar que los puntos geográficos dados tengan el orden adecuado, siendo por ejemplo la latitud superior izquierda mayor que la inferior derecha para ser correcta.

4.2 Módulo de obtención de alturas

La primera necesidad de este módulo es saber cuál es la región de interés del usuario, para lo cual necesita las coordenadas, por lo tanto, lo primero que se realiza en este módulo es la llamada al módulo de interfaz gráfica para recoger dichos datos.

Una vez sabidas las coordenadas se procede a realizar un total de 512 peticiones distintas a la fuente de obtención de alturas, la **Google Elevation API**. El formato de dichas peticiones es el siguiente:

```
url = ELEVATION_BASE_URL + '?' + urllib.parse.urlencode(elvtn_args) + "&key=" + KEY
```

Figura 4-1: Formato de petición a la Google Elevation API

Donde el parámetro **elvtn_args** corresponde con la variable que marca el inicio y el fin de la banda, así como el número de puntos a tomar en dicha banda. El inicio y el fin de la banda se designan con el formato:

latitud_superior_izquierda,longitud|latitud_inferior_derecha,longitud

Donde los valores de latitud son los originales y el de longitud es la variable que va tomando valores distintos en cada banda, de tal manera que se recorre el área por columnas de izquierda a derecha. El número de puntos es tan solo una variable "sample" que he igualado a 512.

Para realizar la obtención de las alturas, hay que tener en cuenta la limitación de 5 peticiones por segundo, la cual nos obliga a vigilar la variable *status* de la respuesta de la API, haciendo un *sleep* de 0.5 segundos antes de proceder con el resto de peticiones en caso de error. Tras la obtención de cada una de las mismas, éstas se guardan en un array que, tras haberse rellenado con todas y cada una de las respuestas válidas, se recorrerá para escribir el archivo JSON en el que se almacenan las respuestas.

Finalmente, tras escribir el archivo JSON del que hará uso el módulo de estructuración de alturas, se pide al usuario el nombre de archivo con el que va a querer guardar la imagen generada. Para esto se vuelve a hacer uso de la librería **easygui**, desplegando un formulario con la misma función que la introducción a mano de las coordenadas. Si el usuario no introduce un nombre, el nombre por defecto será "mapa_de_altura".

4.3 Módulo de estructuración de alturas

Una vez obtenidas las alturas en el archivo JSON, este módulo se encarga de su lectura mediante la librería de Python **json**. Para ello recorro el archivo JSON línea por línea, leyendo cada una de las 512 mediante la función *loads()* de **json**, lo cual convierte cada línea en un diccionario. Este diccionario contiene a su vez 512 diccionarios en el que a el dato que más nos interesa es el que se encuentra tras la clave 'elevation'. Cada una de las elevaciones que vamos leyendo la vamos introduciendo en la matriz de alturas. A la vez que vamos rellenando la matriz, también es necesario ir registrando cuál es la altura mayor y cuál la menor para normalizar todos los valores de la región mapeada.

Una vez generada la matriz, queda normalizarla. Esto es necesario para conseguir una distribución de colores del mapa de altura más detallada, evitando que en el caso de haber valores muy dispares, los colores se repartan de manera desequilibrada.

La normalización la llevo a cabo aplicando min-max con la altura mayor como máximo y la menor como mínimo. Cada valor normalizado así, estará comprendido en un rango del 0 al 1. Para conseguir pues valores entre el 0 y el 255 para poder hacer uso de la escala de grises de los mapas de altura tan solo es necesario multiplicar por 255 cada valor resultante.

4.4 Módulo de generación y conversión de imagen

Este módulo hace uso de la librería de Python **TkInter**, con la que genera una imagen de tamaño 512 píxeles de alto, 512 píxeles de ancho. Antes de comenzar a pintar en el lienzo, realiza una llamada a la función del módulo de estructuración de alturas que genera la matriz de alturas con los valores normalizados. Una vez obtenida dicha matriz, tan solo queda recorrer cada píxel de la imagen tomando su valor de la conversión a color hexadecimal de cada posición de la matriz, esto es, para la posición X, Y de la imagen, llamaremos a una función que he llamado *getHexColor* la cual recibe como argumento el valor de la matriz en la posición X, Y. Esta función nos devolverá el código hexadecimal en función del valor decimal recibido. Este código es básicamente la traducción a hexadecimal del valor decimal con el cuidado de que la función usada para ello, hex(), escribe delante del valor la x que marca su base y que, si es un número de un solo dígito, habrá que añadir manualmente el 0 delante para que sea correctamente interpretado por el código RGB que, como ya hemos hablado antes, tiene, por ejemplo para el color blanco, el código #FFFFFF. Nosotros de un único número entre 0 y 255 podemos tan solo sacar una cifra desde 0x00 a 0xFF, por lo que será necesario triplicarla para cada espectro de color.

Una vez generada la imagen solo resta guardarla bajo el nombre escogido por el usuario en formato png y ejecutar el comando de **ImageMagick** pertinente para su conversión al formato raw.

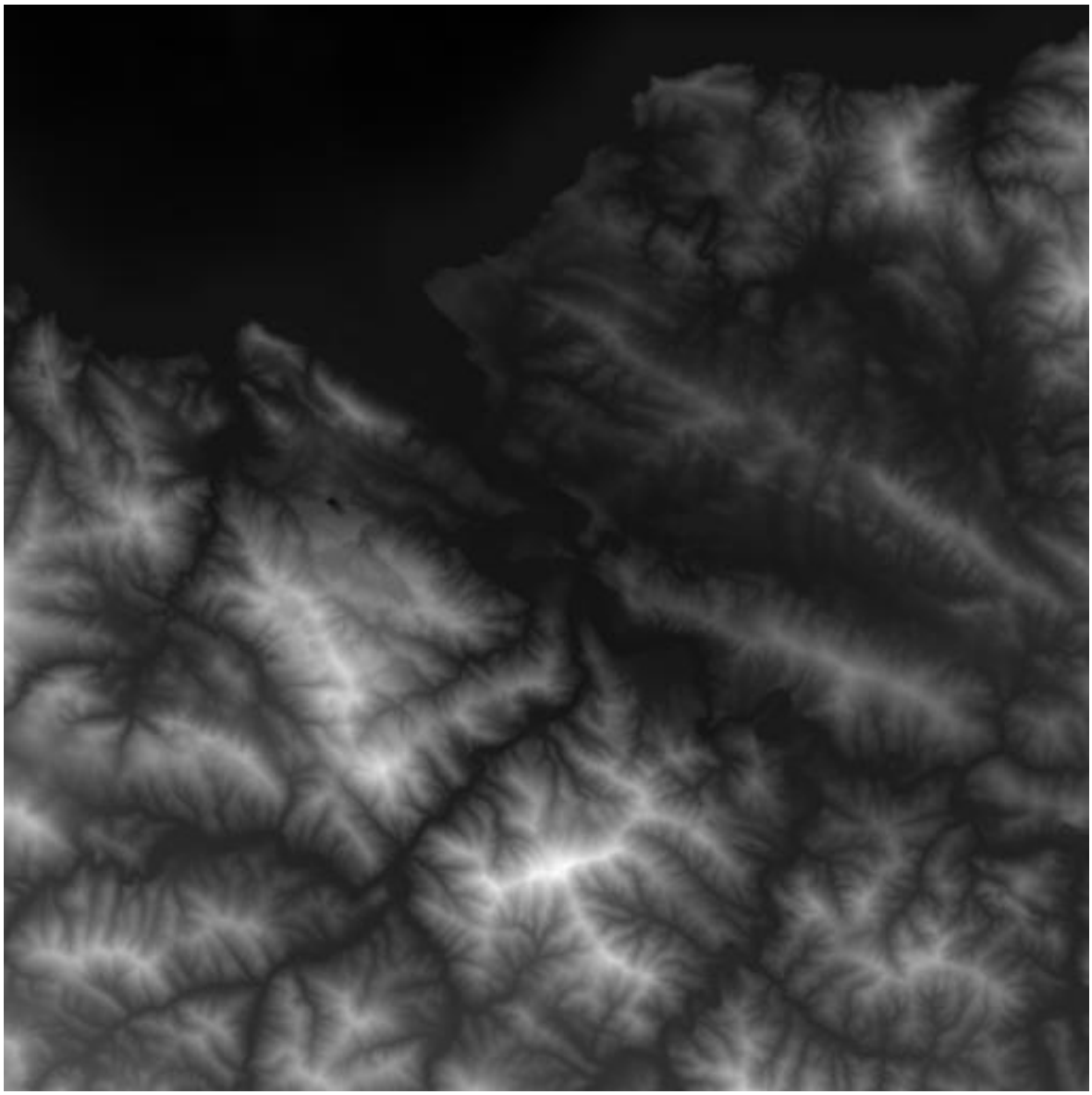

Figura 4-2: Mapa de altura de Bilbao generado

5 Integración, pruebas y resultados

Una vez descrito el diseño y desarrollo de cada módulo queda explicar las interacciones entre los mismos, así como las pruebas y resultados de su funcionamiento conjunto al ejecutar la aplicación.

5.1 Integración

La interacción entre los distintos componentes sigue pues la siguiente lógica. Primero el usuario ha de poder obtener las coordenadas con las que hacer referencia al área en el que está interesado, para ello puede hacer uso de la aplicación JavaScript propuesta o de cualquier otro medio a disposición de este, como por ejemplo **Google Maps**. Tras esto todo está listo para la ejecución de la aplicación Python. Como ya hemos descrito, la aplicación requerirá al usuario la introducción de las coordenadas que describan la región de interés, tras esto tomará el control el módulo de obtención de alturas, el cual realizará el particionado del área en una matriz de 512 por 512 y realizará las peticiones oportunas a la **Google Elevation API**. Tras recabar todas las alturas, éstas se escriben en un archivo JSON, listas para que el siguiente módulo las estructure en una matriz de valores aplicando la normalización que establezca los valores en un rango entre 0 y 255. Con dicha matriz tan solo falta pintar una imagen de 512 píxeles por 512 píxeles en una escala de grises, con cada tonalidad descrita por el valor entre 0 y 255 de la matriz de alturas. Una vez pintada la imagen tan solo queda guardarla bajo el nombre dado por el usuario y, gracias al programa **ImageMagick**, convertirla al formato raw, lista para ser importada a Unity.

5.2 Pruebas

Las pruebas realizadas para comprobar el correcto funcionamiento de la herramienta han comprendido desde la correcta toma de datos por parte del mapa interactivo, pasando por la comunicación con la API de Google, hasta la generación de distintas localizaciones como mapas de altura y su reconstrucción mediante Unity para estudiar la diferencia de precisión dependiendo de las dimensiones del área escogida.

5.3 Resultados

Ejemplos de las localizaciones representadas en mapas de altura son, Roma, Ceberio, el país completo de Ecuador y una pequeña sección de Alcobendas. Estos ejemplos han sido escogidos debido a que cubren bien las diferencias de magnitud del área, así como de cotas de altura. Estas pruebas constan de tres imágenes, la fotografía satélite de Google Maps, el mapa de altura generado con la herramienta desarrollada y por último la representación del terreno 3D en Unity. El procedimiento seguido para las mismas se encuentra detallado en el manual de usuario situado en el anexo. Los resultados son los siguientes:

Roma con las coordenadas:

- Arriba izquierda = (41.947247, 12.454552)
- Abajo derecha = (41.918037, 12.504092)

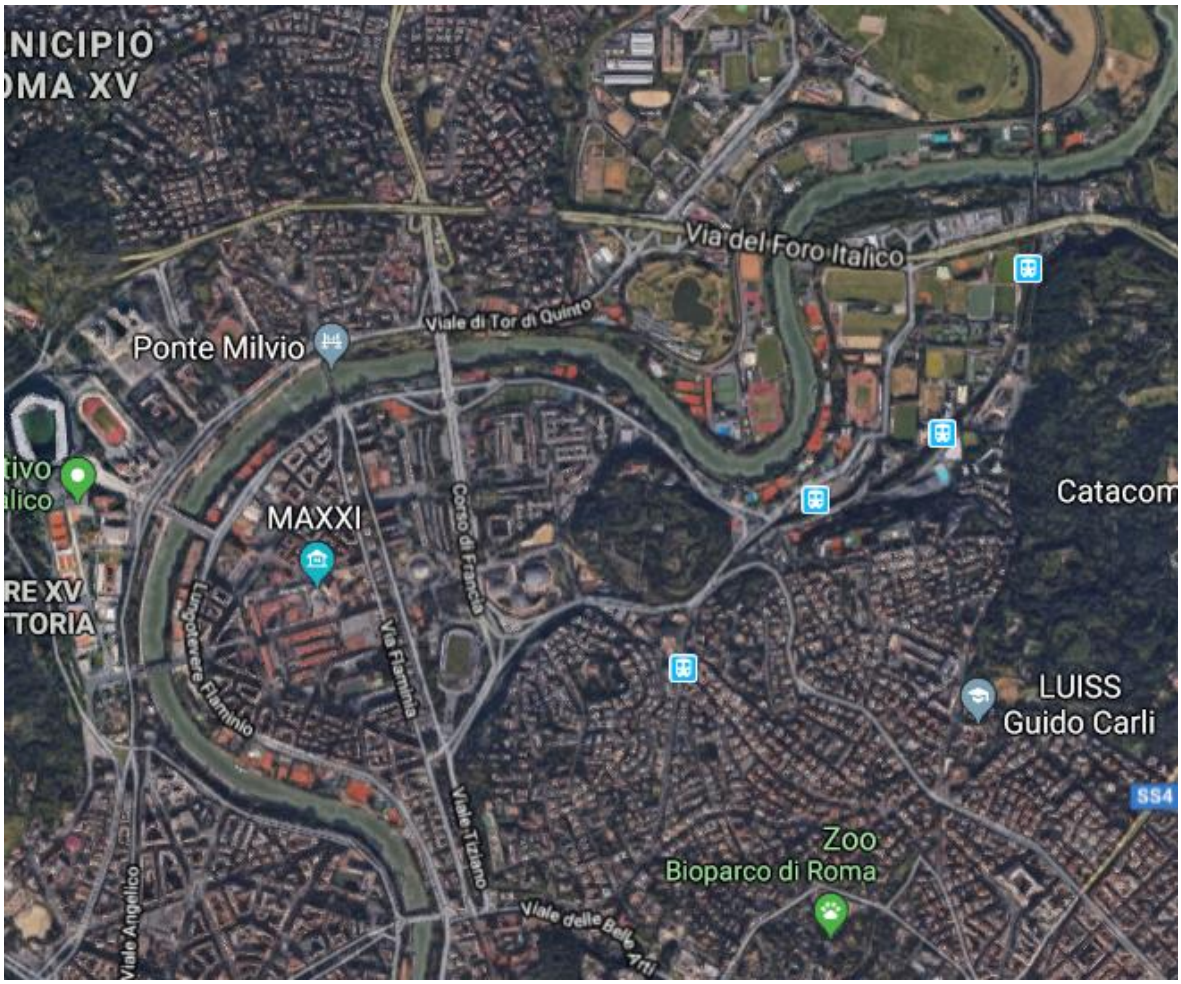

Figura 5-1: Sección de Roma desde Google Maps.

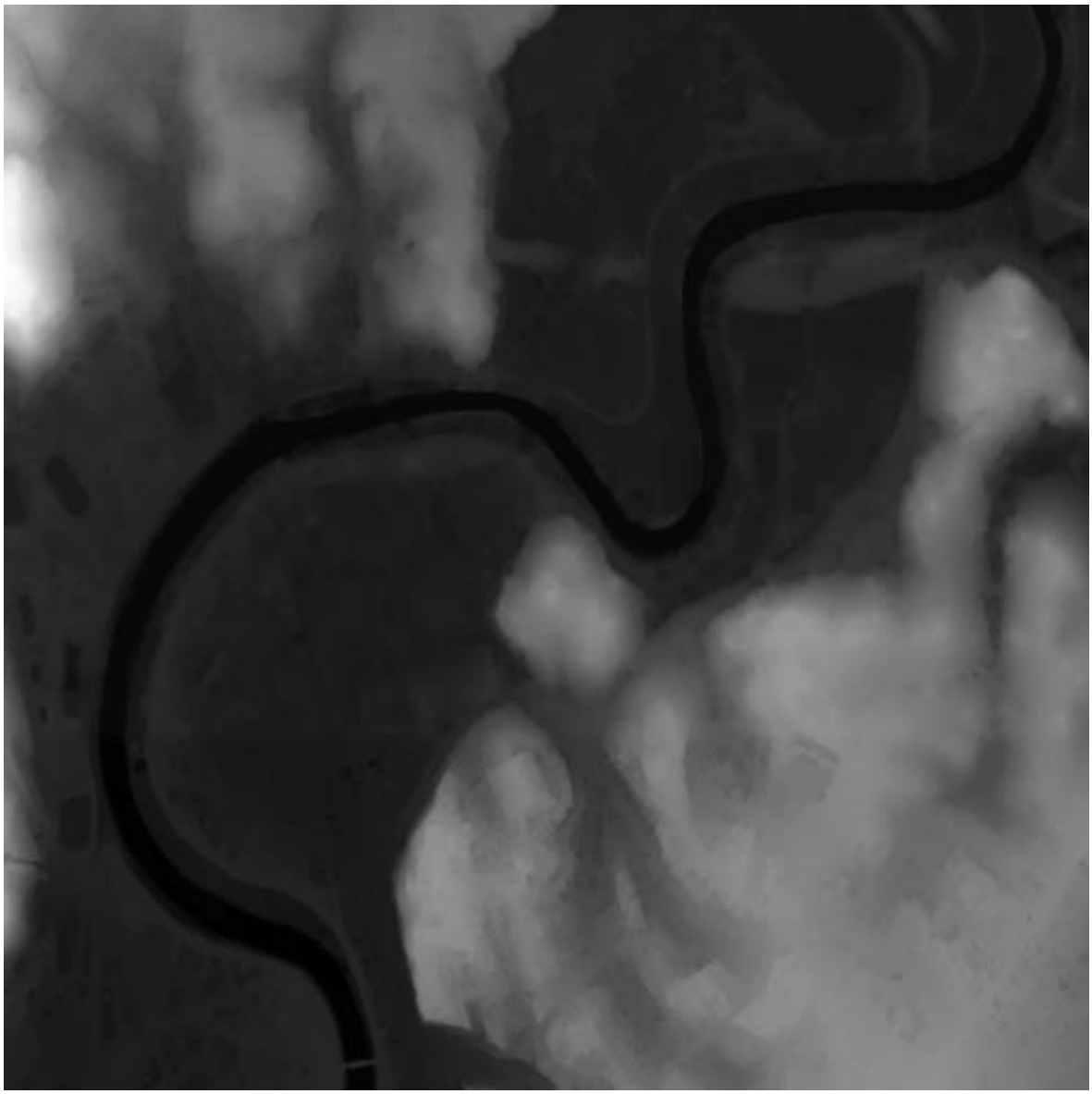

Figura 5-2: Mapa de altura de Roma generado

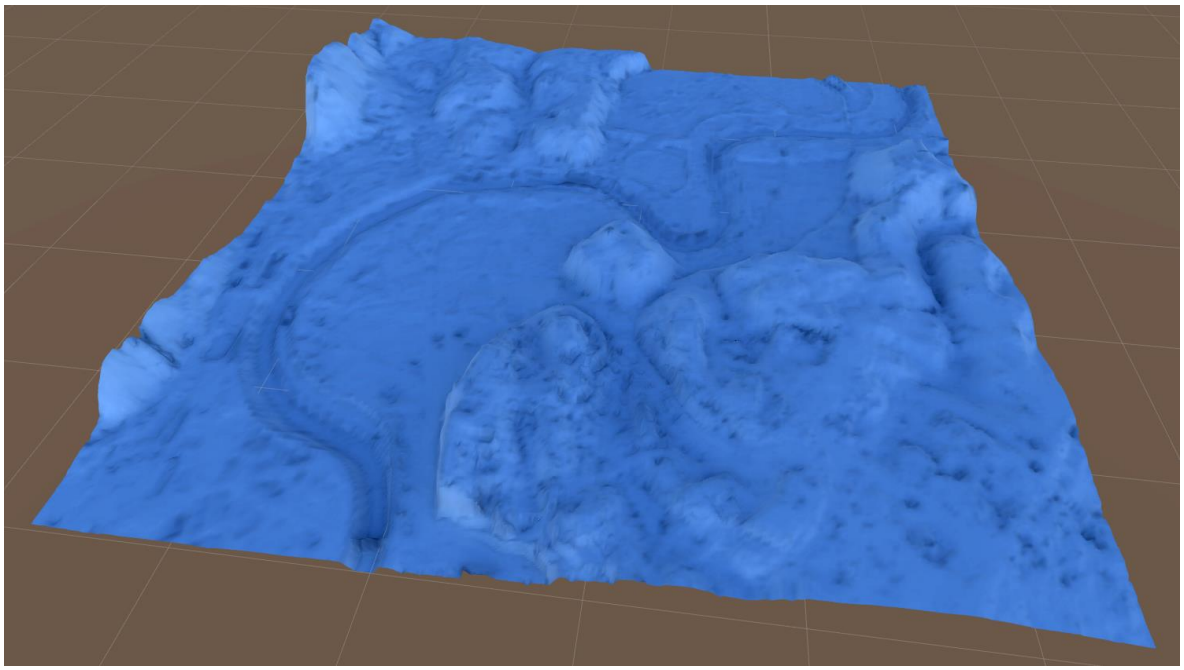

Figura 5-3: Modelo 3D de Roma.

En la representación se puede distinguir claramente el surco excavado por el río Tíber, así como el terreno abrupto de la zona cercana, como vemos con bastante detalle. Como se ha explicado previamente, cuanto más oscuro es el tono más baja es la altura interpretada, cuanto más clara, más alta.

Ceberio con las coordenadas:

- Arriba izquierda = (43.194922, -2.909769)
- Abajo derecha = (43.107182, -2.799282)

Figura 5-4: Sección de Ceberio desde Google Maps.

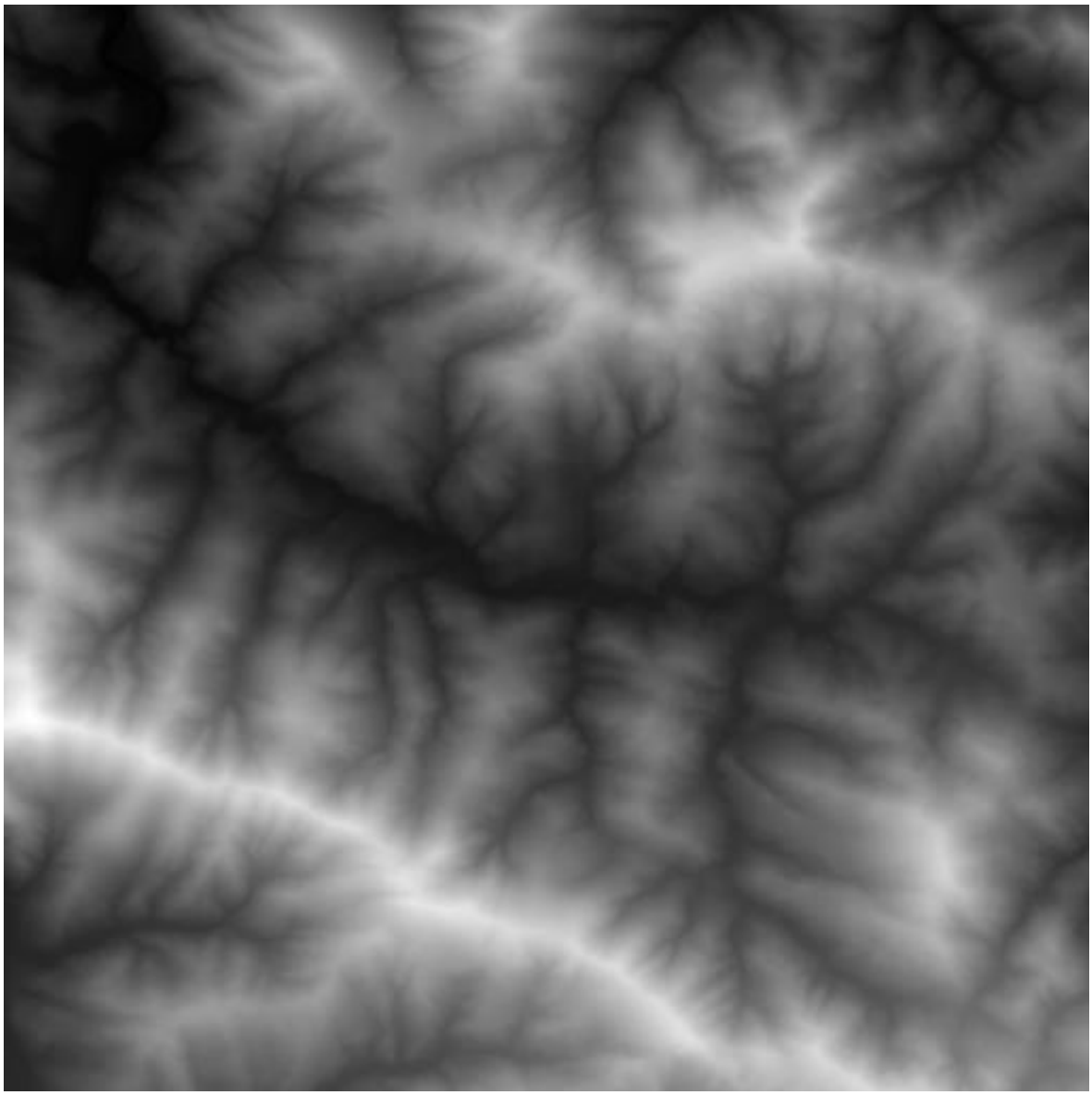

Figura 5-5: Mapa de altura de Ceberio generado

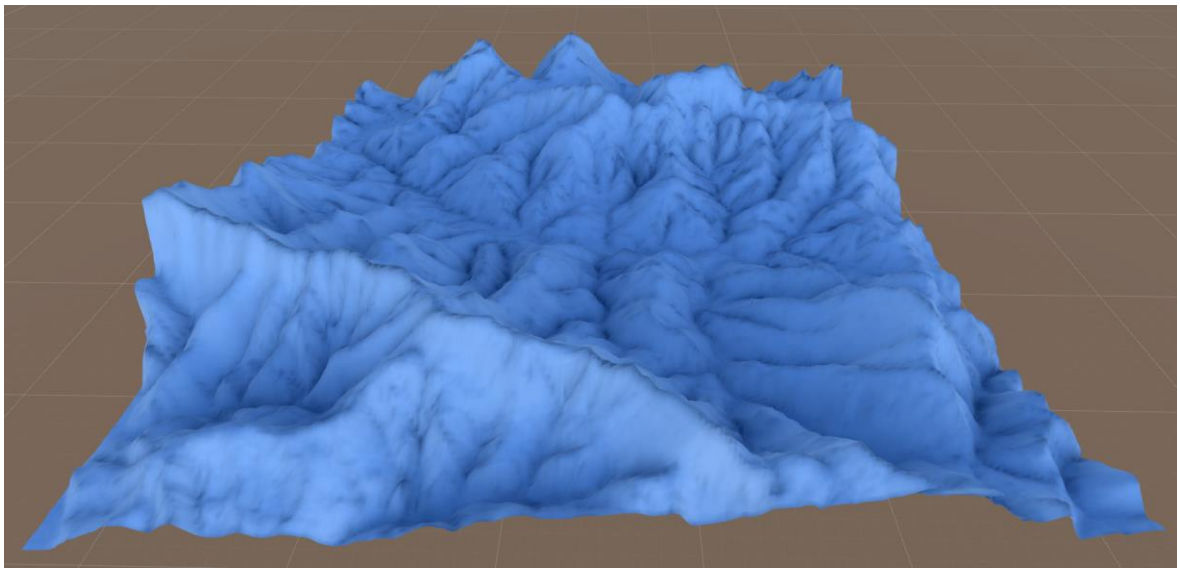

Figura 5-6: Representación 3D en Unity de Ceberio.

Como podemos apreciar, los resultados para áreas geográficas de dimensiones medias como ciudades o montañas son bastante precisas, obteniendo una gran calidad en el mapa que permite incluso reconocer el área representada. A continuación comprobaremos su precisión en un área más extensa.

Ecuador con las coordenadas:

- Arriba izquierda = (1.779499, -82.727050)
- Abajo derecha = (-5.287887, -75.651855)

Figura 5-7: Ecuador desde Google Maps.

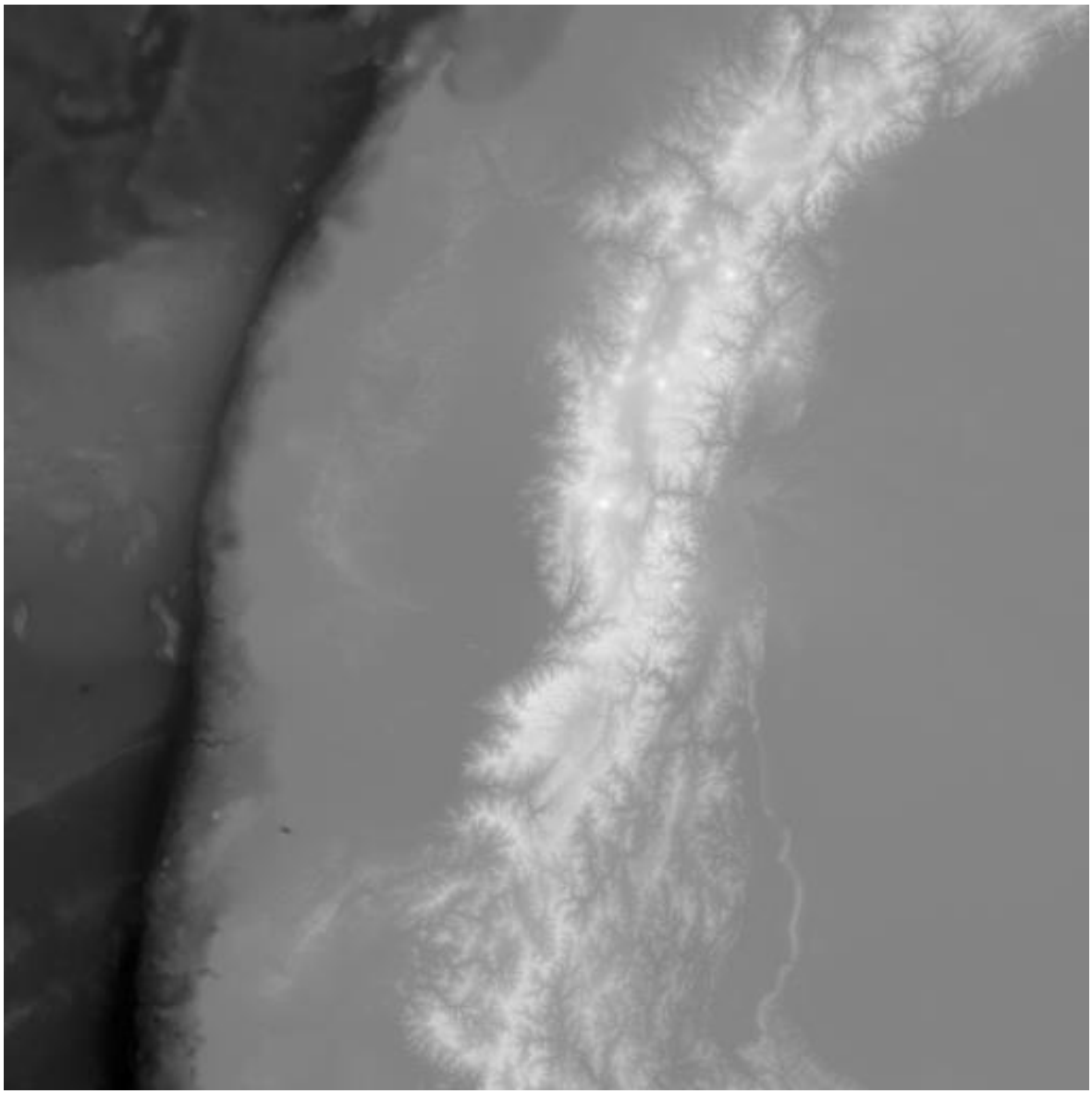

Figura 5-8: Mapa de altura de Ecuador generado

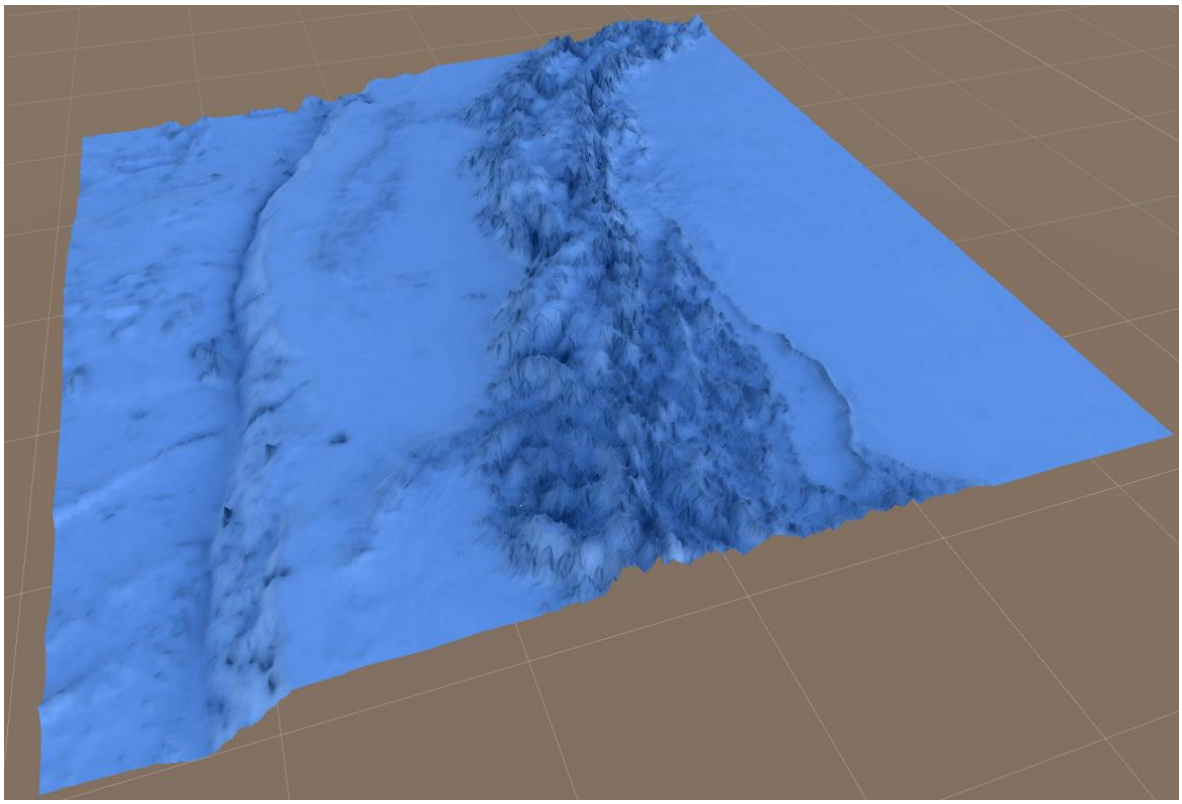

Figura 5-9: Representación 3D en Unity de Ecuador

Al escoger áreas de mayor extensión, como vemos, se sigue defendiendo bien, a costa de una precisión a nivel medio, ya que se centra en las principales características del terreno (la fosa de Atacama y los Andes).

Sección de Alcobendas con las coordenadas:

- Arriba izquierda = (40.545928,-3.663189)
- Abajo derecha = (40.539748, -3.655056)

Figura 5-10: Sección de Alcobendas desde Google Maps

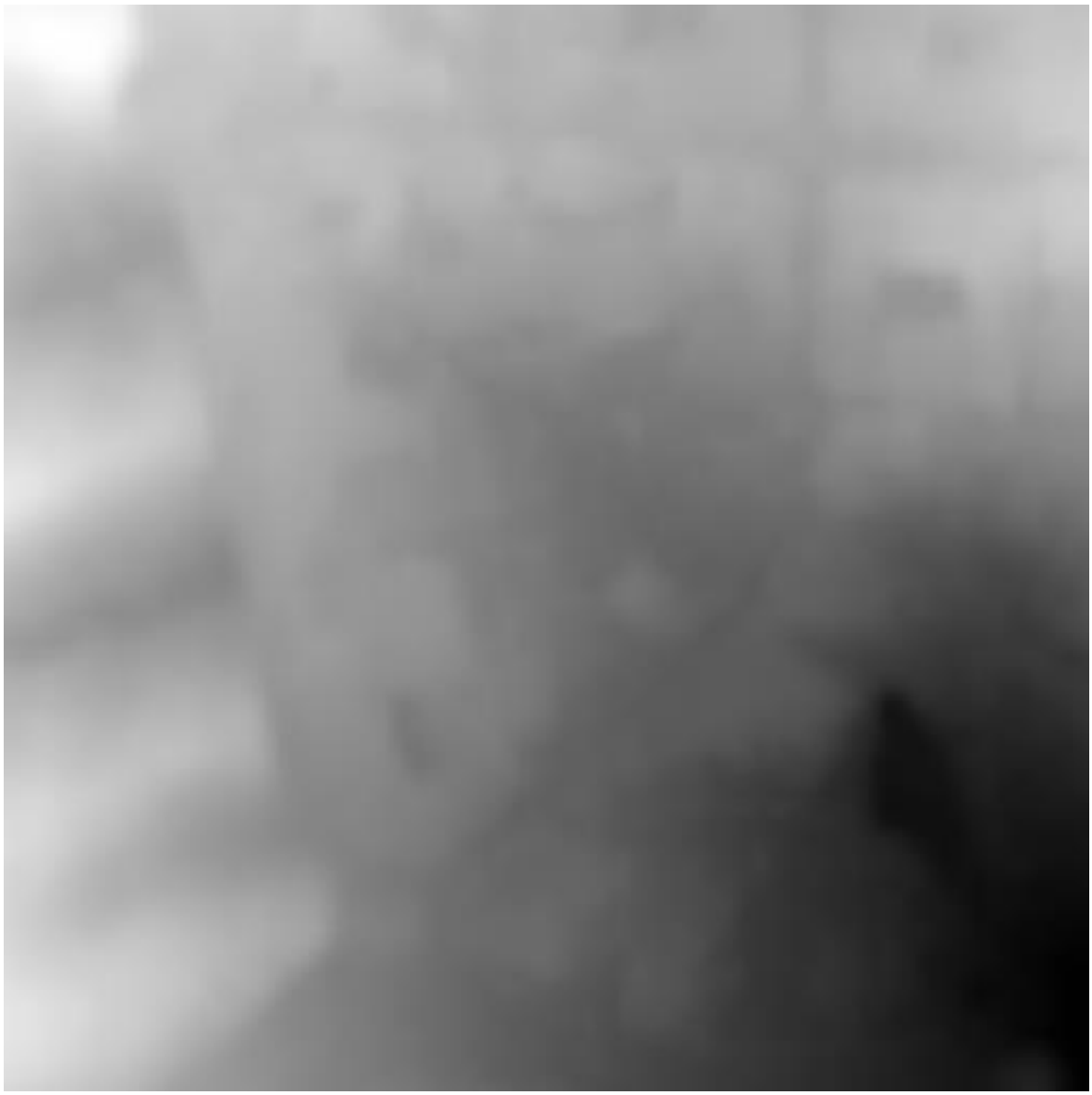

Figura 5-11: Mapa de altura sección de Alcobendas generado

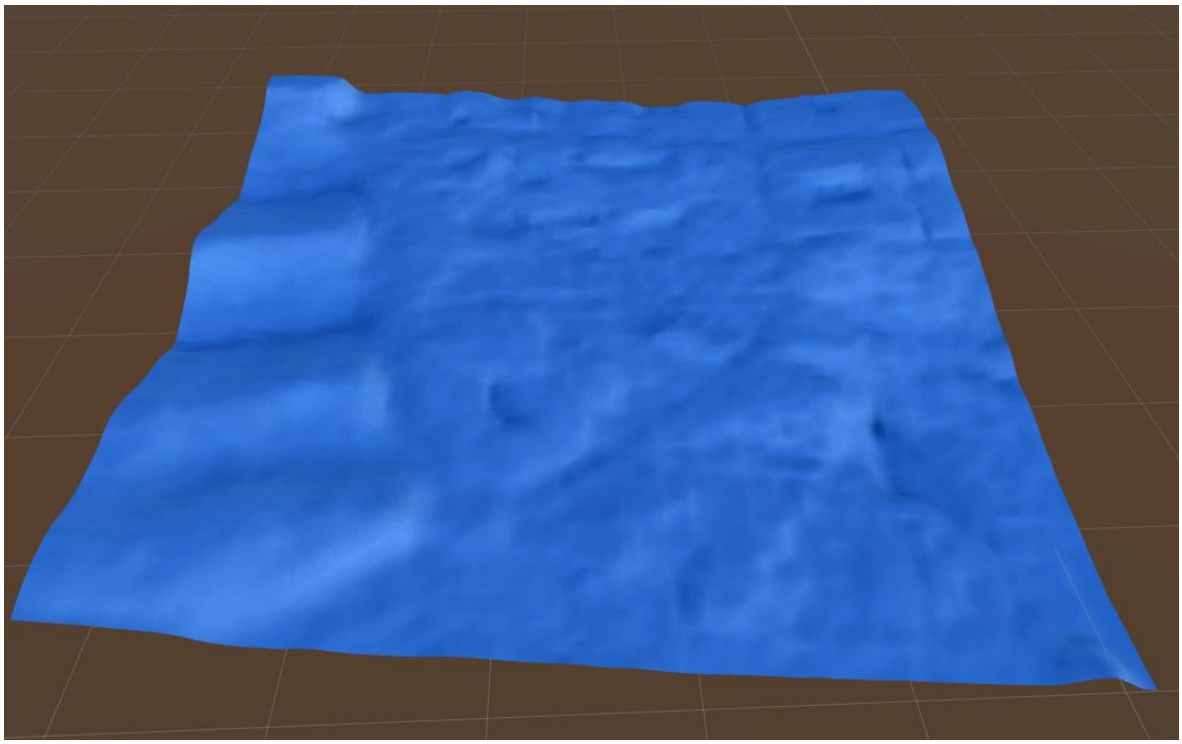

Figura 5-12: Representación 3D de la sección de Alcobendas en Unity

Como podemos observar, al tomar un área de aproximadamente un kilómetro con una diferencia de altura desde el punto más bajo al más alto de cuarenta y tres metros, el mapa de altura es más difícilmente interpretable a ojo, aun así, el resultado de la interpretación del mapa por parte de Unity cumple las expectativas correctamente. Si nos fijamos con atención podemos llegar a detectar las calles, así como alguna que otra urbanización, pero ya con una pérdida de detalle notable con respecto a los ejemplos previos.

En definitiva, podemos sacar en conclusión que la herramienta es bastante flexible, pudiendo adecuarse a las distintas magnitudes que se le quieran requerir. De todas formas, como se ha demostrado, la herramienta muestra su mayor potencial con áreas no demasiado reducidas, donde puede mostrar una mayor distinción entre los distintos detalles.

6 Conclusiones y trabajo futuro

6.1 Conclusiones

El estudio de la generación de terrenos 3D mediante la creación de mapas de altura me ha enseñado lo útil que puede llegar a ser una aplicación con estas características. Aun teniendo el hándicap de la obtención de las alturas por medio de un servicio restringido en su versión gratuita, la posibilidad de generar este tipo de representaciones no es tan inaccesible, y sus usos son múltiples, desde la generación del mundo en los videojuegos o por ejemplo en simuladores de vuelo hasta la creación de mapas tridimensionales para tal vez en un futuro la realidad virtual o aumentada. En cuanto al producto hasta el momento, puedo decir que estoy contento con la herramienta que he desarrollado, pues la calidad de representación es mayor de lo que esperaba. El proyecto en general me ha enseñado lo importante que es una buena organización para afrontar un reto de estas dimensiones, he aprendido la importancia de investigar y diseñar la aplicación con cuidado, habiendo sido esto el cuerpo del iceberg sobre el que se sustenta la aplicación final, que es lo que sobresale a simple vista.

6.2 Trabajo futuro

Para extender más allá la calidad del proyecto se me ocurre completarlo con un par de ideas enfocadas a aumentar su alcance, así como su accesibilidad, como por ejemplo su integración en Unity como herramienta o encapsularla en una aplicación web con un framework como por ejemplo Django, otorgando así más cohesión y comodidad al usuario final. También sería conveniente encontrar una fuente de obtención de alturas sin las limitaciones de pago que tiene la actual. Para aumentar la especialización de la herramienta tengo en mente también decorar el terreno generado automáticamente con distintas texturas dependiendo de su localización geográfica de forma autónoma e inteligente, dando por ejemplo un tono verde a las laderas de una montaña y tonos marrones a su pico.

Referencias

[1] Mike Minotti, “Cuphead has sold over 3 million copies”, de VentureBeat, 9 de agosto de 2018, https://venturebeat.com/2018/08/09/cuphead-has-sold-over-3-million-copies/

[2] Megan Farokhmanesh, “The Binding of Isaac has sold more than two million copies to date”, de Polygon, 20 de abril de 2013, https://www.polygon.com/2013/4/20/4246254/the-binding-of-isaac-has-sold-more-than-two-million-copies-to-date

[3] Ian Boudreau, “Slay the Spire has sold more than 1.5 million copies”, de PCGamesN, 19 de marzo de 2019, https://www.pcgamesn.com/slay-the-spire/slay-the-spire-sales

[4] “Factorio”, steamspy, https://steamspy.com/app/427520

[5] Leaflet – an open-source JavaScript library for mobile friendly interactive maps, https://leafletjs.com/index.html

[6] “Modelado 3D”, Wikipedia, https://es.wikipedia.org/wiki/Modelado_3D

[7] “Unity (game engine), Wikipedia, https://en.wikipedia.org/wiki/Unity_(game_engine)]

[8] Richard Baxter, “Streaming OSM/Terrain Unity Package”, https://sites.google.com/site/raxterbaxter/projects/current-projects/streaming-osm-terrain-unity-package

[9] “API vo.6”, de OpenStreetMap Wiki, https://wiki.openstreetmap.org/wiki/API_v0.6, 4 de junio de 2019

[10] “SRTM Digital Elevation Data Version 4”, de Earth Engine Data Catalog, https://developers.google.com/earth-engine/datasets/catalog/CGIAR_SRTM90_V4

[11] “easygui API”, de easygui, 20 de diciembre de 2014, http://easygui.sourceforge.net/api.html

[12] “Developer Guide”, Google Maps Platform, 1 de abril de 2019, https://developers.google.com/maps/documentation/elevation/intro

[13] “Convert”, de ImageMagick, https://imagemagick.org/script/convert.php

[14] “TkInter – Python Wiki”, de Python, https://wiki.python.org/moin/TkInter

Glosario

API	Application Programming Interface, Código de comunicación entre aplicaciones.
JSON	JavaScript Object Notation, es un formato ligero de intercambio de datos.
Unity	Motor de videojuego multiplataforma programado en C, C++ y C# creado por Unity Technologies.
Indie	Abreviatura de independiente, aplicado a los videojuegos hace referencia a los estudios pequeños, con presupuestos humildes que tienden a centrarse en la innovación.
Python	Es un lenguaje de programación interpretado de alto nivel cuya filosofía gira en torno a la legibilidad.
TkInter	Librería de Python sobre interfaces graficas de usuario.
C#	Lenguaje de programación orientado a objetos desarrollado por Microsoft.
JavaScript	Es un lenguaje de programación interpretado de alto nivel. Es uno de los principales núcleos de la World Wide Web.
Django	Es un framework web basado en Python, que ofrece un modelo vista controlador con múltiples herramientas que agilizan el desarrollo de aplicaciones web.
Escala de grises	Paleta de colores que toma para cada píxel el mismo valor en los espectros RGB, tomando tonalidades por lo tanto grisáceas desde el negro al blanco.
ImageMagick	Software gratuito para el tratamiento de imágenes.
Normalización min-max	Método de normalización que transforma un conjunto de valores a un rango entre 0 y 1 de manera proporcional a su valor original.
Sleep	Función de distintos lenguajes de programación. Pausa la ejecución del código durante un intervalo de tiempo especificado.
.png	Portable Network Graphics, formato de imagen popular que hace uso de un sistema de compresión sin pérdidas de información.
.raw	Del inglés “crudo”, es un formato de imagen con un procesamiento mínimo que respeta los datos capturados por la cámara.

Framework	Entorno de trabajo orientado a un lenguaje que permite cohesionar los distintos elementos de un proyecto con unas directrices propias.
Asset	Recurso o ítem de Unity. Por ejemplo, una textura un objeto o un script.

Anexos

A Manual de usuario

Los pasos a seguir para generar un mapa de altura de cualquier parte del globo con esta aplicación son los siguientes:

Obtener las coordenadas:

1. Abrir el mapa interactivo “MapaSeleccionCoordenadas.html”.

2. Una vez abierto se mostrará el mapamundi centrado en la escuela politécnica superior.

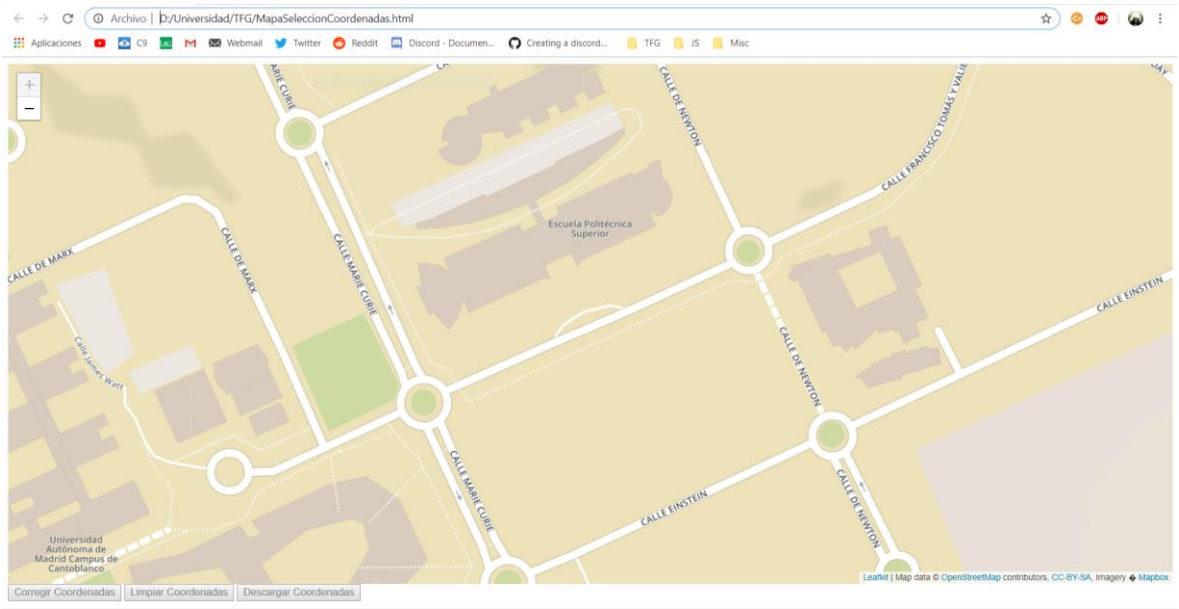

Figura 0-1: Mapa interactivo centrado en la EPS.

3. Desplazarse hasta la zona deseada haciendo scroll con la rueda del ratón o ampliando con el panel táctil para alejar la vista y pinchando y arrastrando con el botón primario para desplazarse.

4. Una vez localizada el área deseada hacer clic con el botón primario sin arrastrar para colocar el primer marcador.

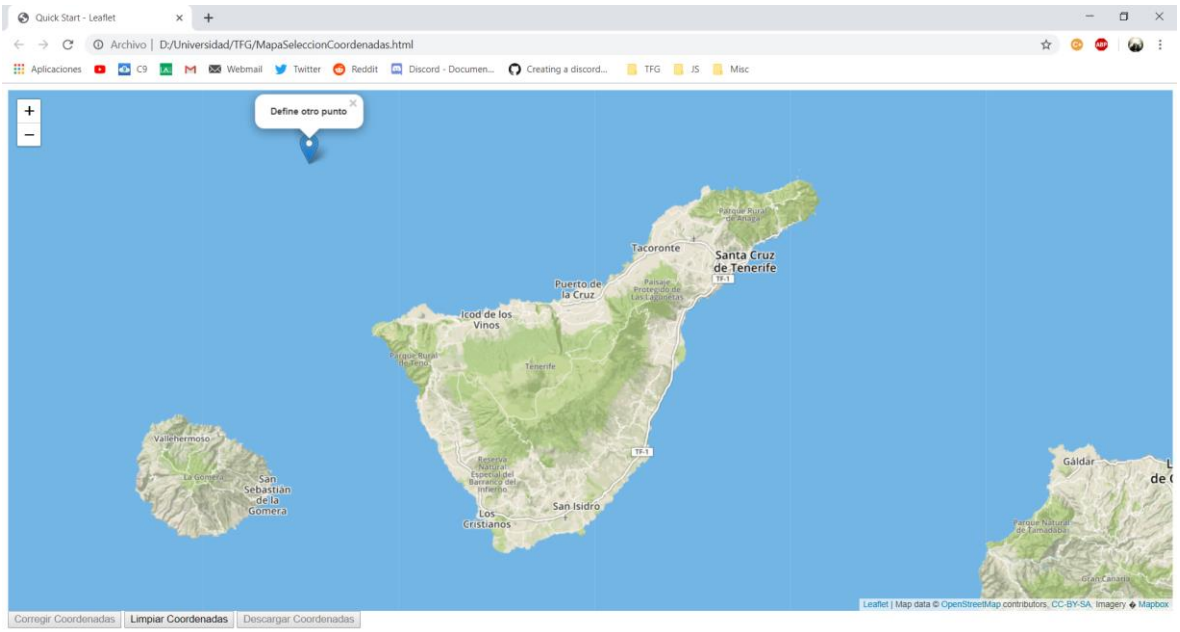

Figura 0-2: Mapa interactivo con un marcador cerca de Tenerife.

5. Hacer clic de nuevo para marcar la otra esquina que denota el área a escoger.

    a. En caso de error presionar el botón “Limpiar Coordenadas”.

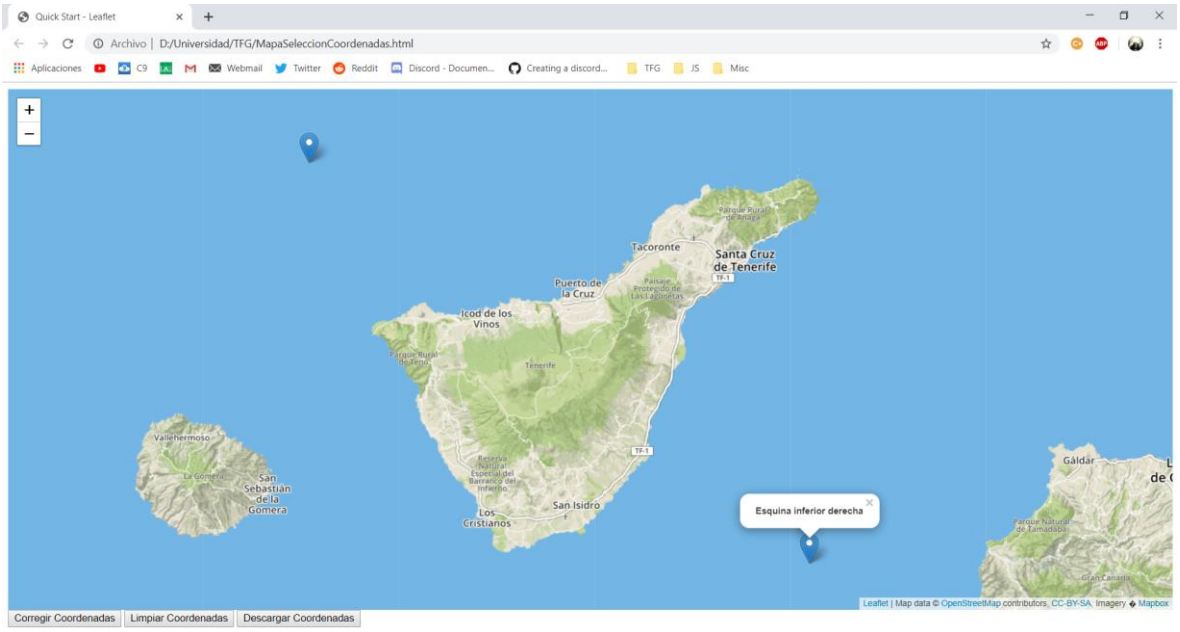

Figura 0-3: Mapa interactivo con dos marcadores rodeando Tenerife.

6. Presionar el botón “Corregir Coordenadas” (Este paso se puede omitir, pero es aconsejable para obtener un mapa de altura con la mayor calidad posible).

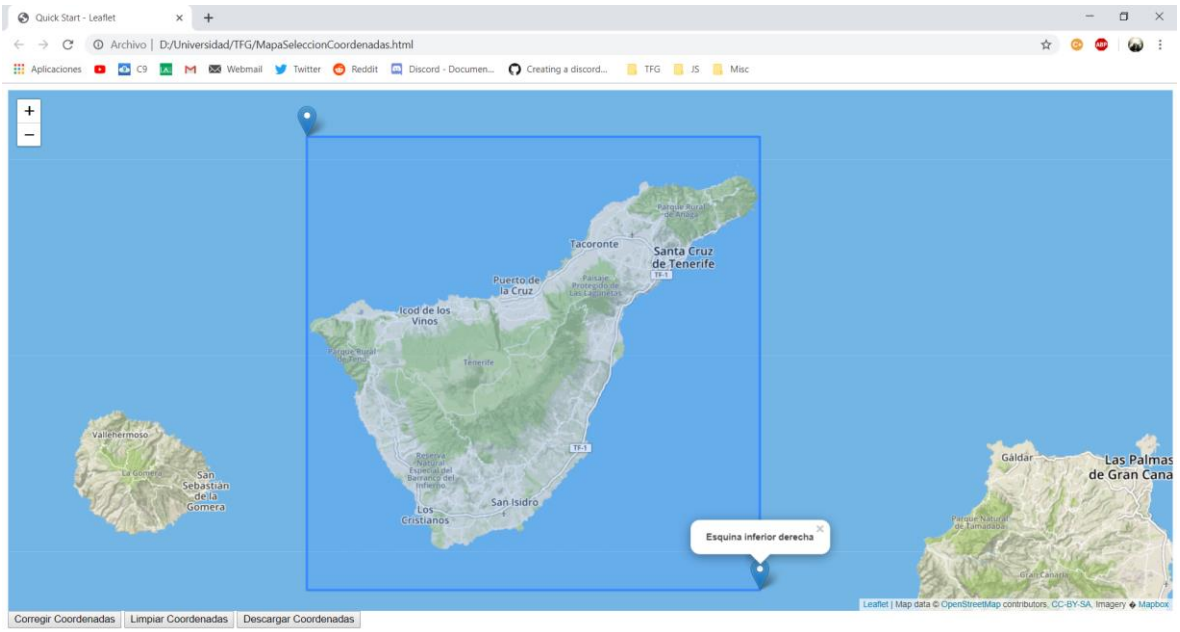

Figura 0-4: Mapa interactivo con el área de Tenerife seleccionada.

7. Presionar el botón “Descargar Coordenadas”.

8. Introducir el nombre deseado para el archivo de las coordenadas.

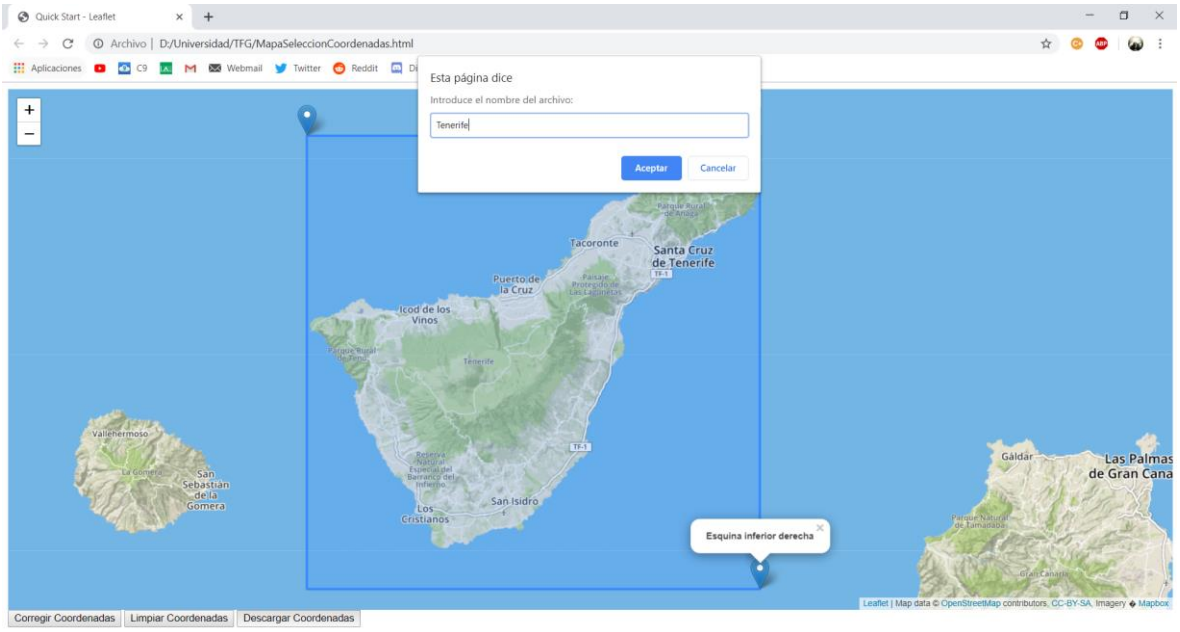

Figura 0-5: Mapa interactivo con el desplegable de la descarga para seleccionar el nombre.

9. Una vez hecho esto deberíamos tener descargado el archivo “Tenerife.txt” con el siguiente contenido:

28.649620345339766,-16.929931640625004,27.9337540167772,-16.11694335937500

Generación del mapa de altura:

1. Ejecutar la aplicación (python Main.py desde terminal en generadorHeighmaps/src).

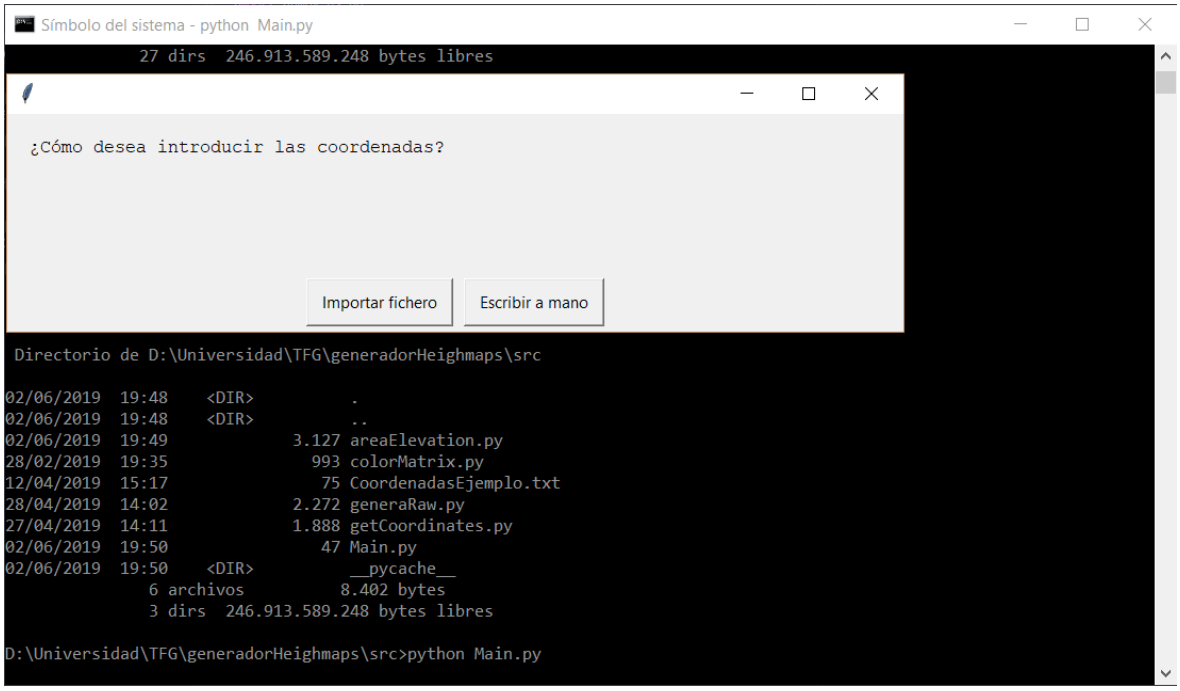

Figura 0-6: Interfaz de la aplicación, selección de introducción de coordenadas.

2. Introducir las coordenadas:
    a. Importando un fichero (por ejemplo Tenerife.txt)

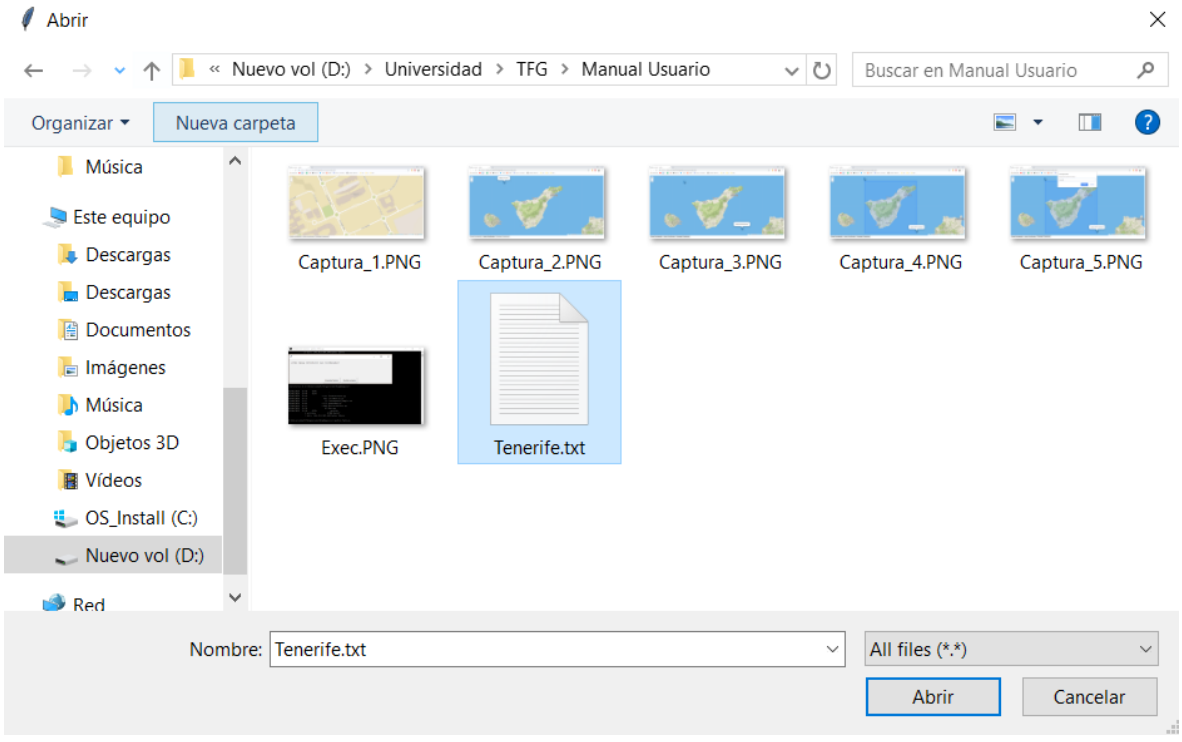

Figura 0-7: Importación de archivo con las coordenadas.

b. Escribiéndolas a mano.

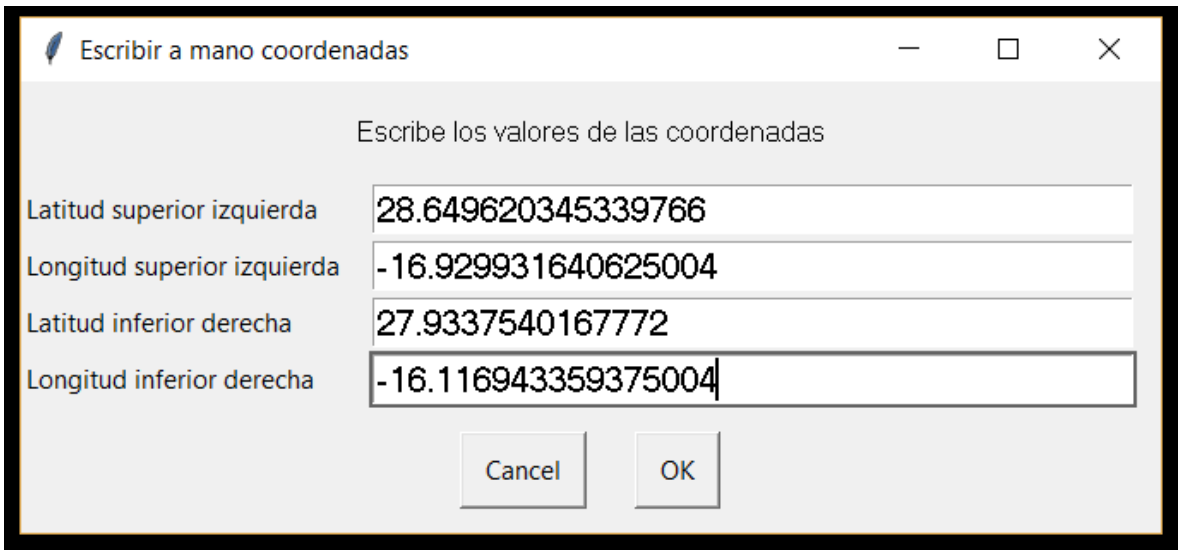

Figura 0-8: Introducción de las coordenadas a mano.

3. Escoger el nombre de la imagen generada.

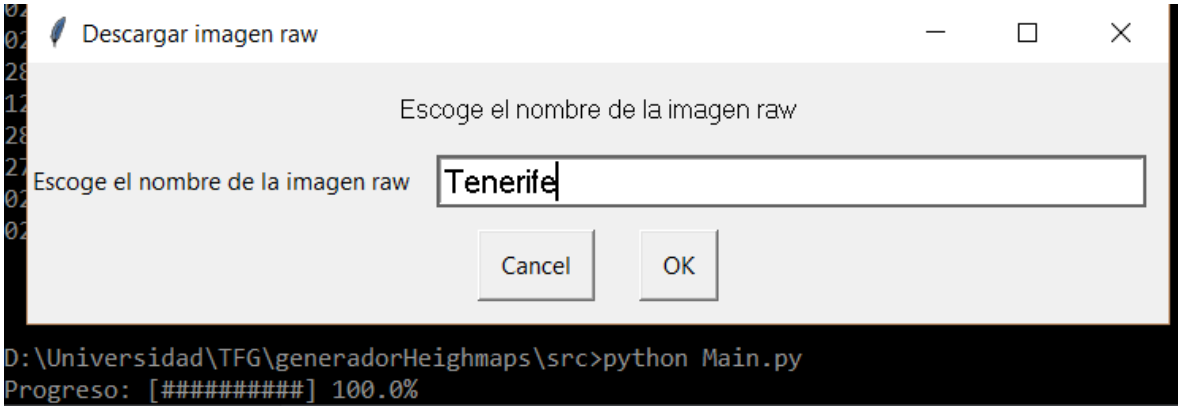

Figura 0-9: Selección del nombre del mapa a descargar.

4. Tras procesar al completo los datos de la imagen se mostrarán por pantalla informaciones sobre las diferencias de altura en dicho terreno y tendremos nuestro mapa de altura listo en la carpeta generadorHeighmaps/output/ tanto en formato .png como en .raw.

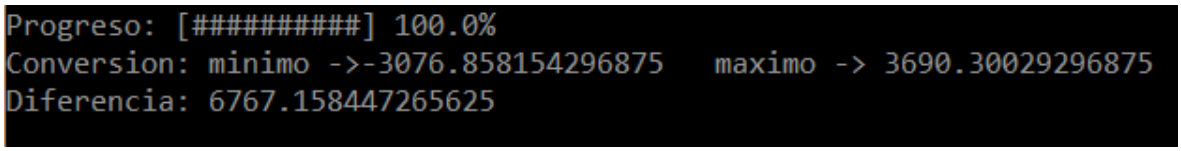

Figura 0-10: Información por terminal de la aplicación al procesar Tenerife.

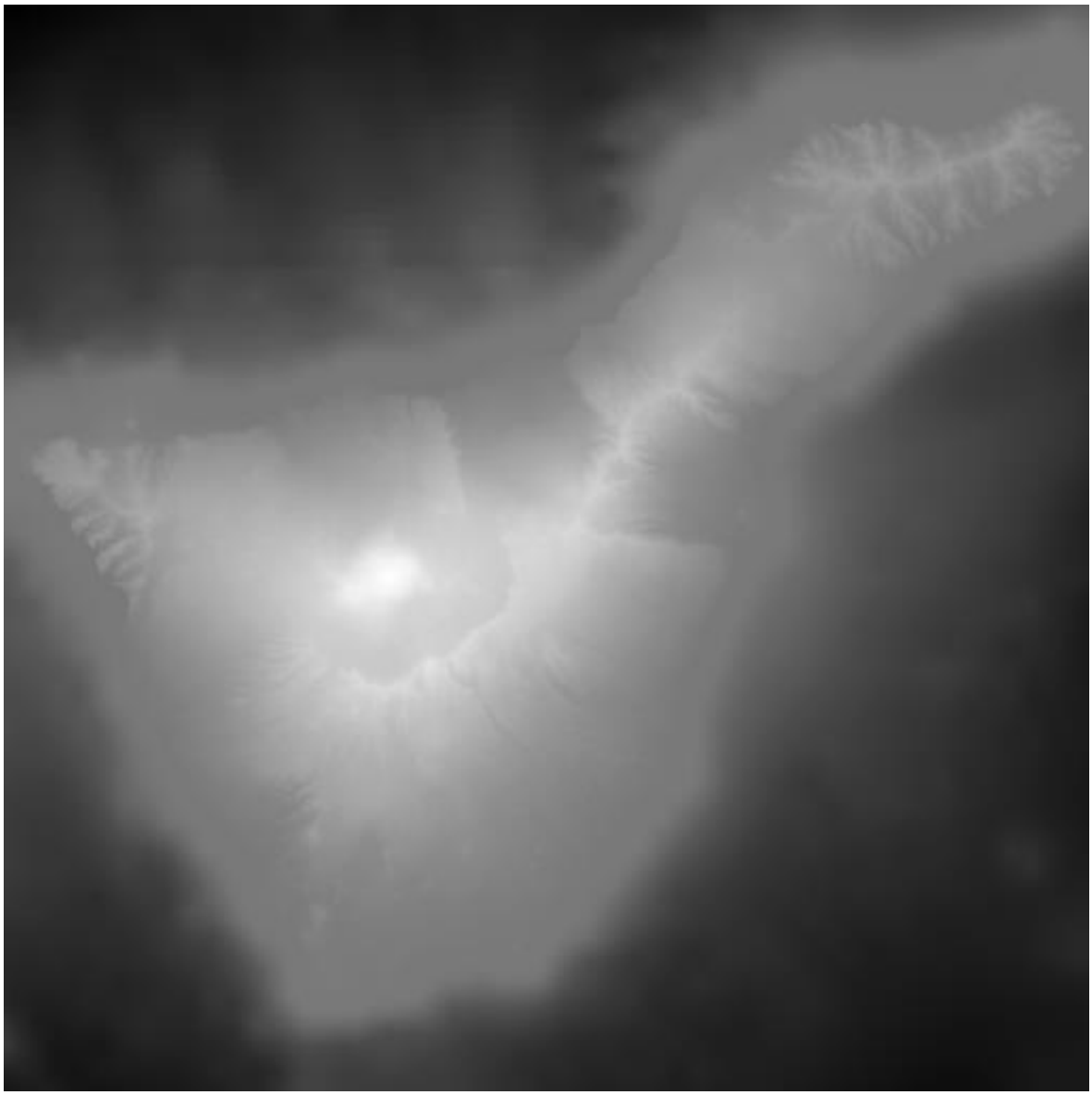

Figura 0-11: Mapa de altura generado. En este caso Tenerife.

Importación y uso en Unity:

Partiendo de un proyecto abierto, para generar el terreno deseado a partir del mapa de altura es necesario:

1. Generar el terreno plano haciendo clic derecho en la escena, 3D Object > Terrain al que podremos renombrar desde la pestaña de Inspector (en este ejemplo a Tenerife).

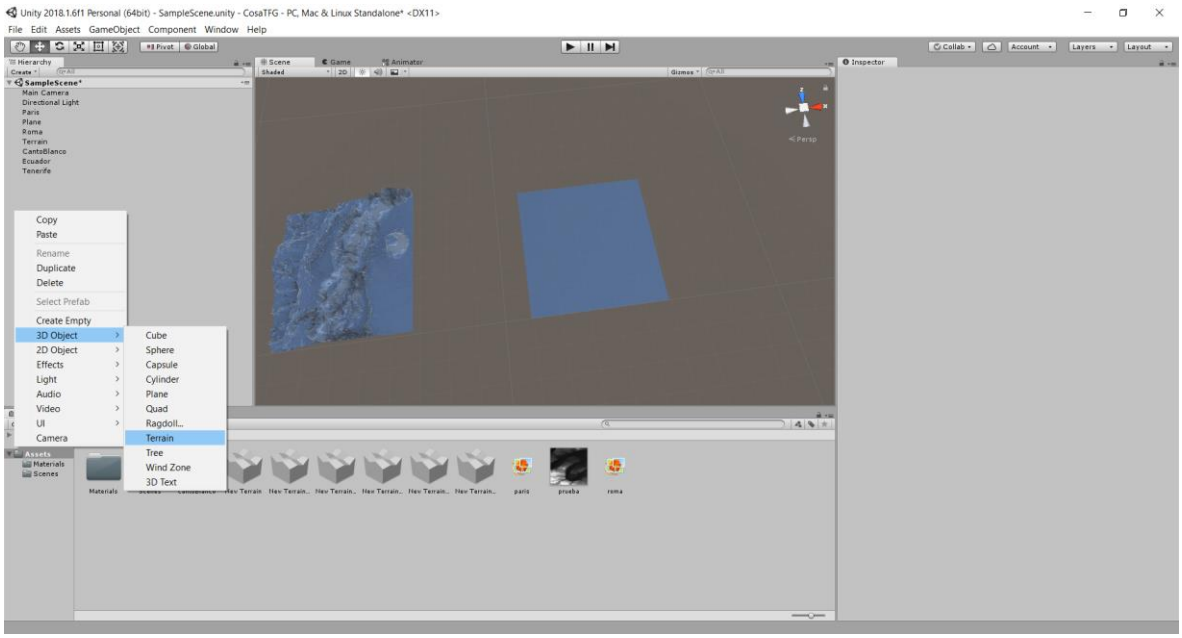

Figura 0-12: Creación en Unity de un objeto de tipo terreno.

2. Desde la ventana de Inspector del terreno, más concretamente en la pestaña Terrain Settings, presionar el botón Import Raw y seleccionar nuestro mapa de altura, para este ejemplo Tenerife.raw.

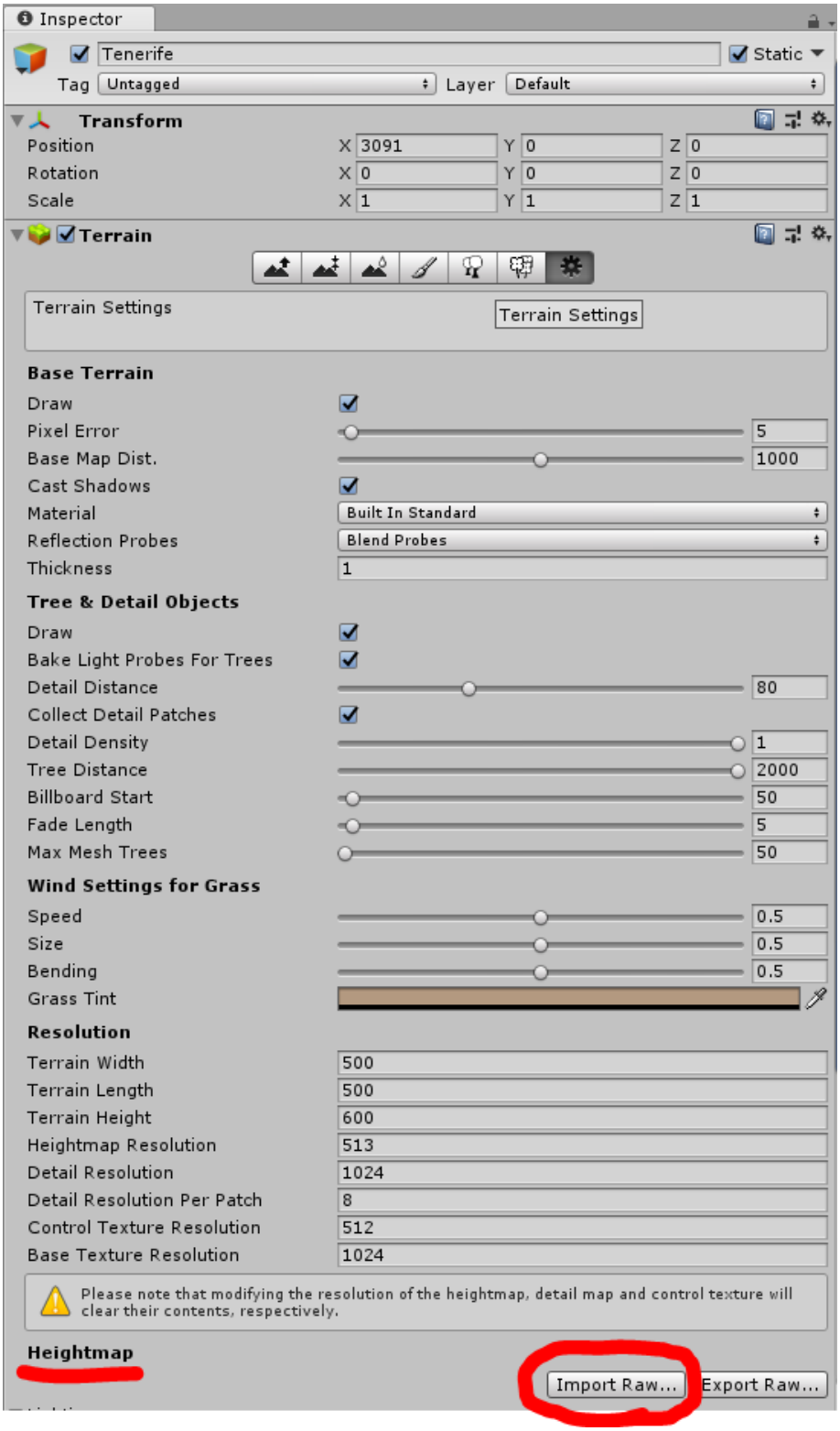

Figura 0-13: Ventana inspector del terreno creado en Unity

3. Una vez importado, como se puede apreciar, ya se ha moldeado el terreno conforme al mapa de altura, a partir de aquí todo queda en manos del usuario para adaptarlo a su gusto.

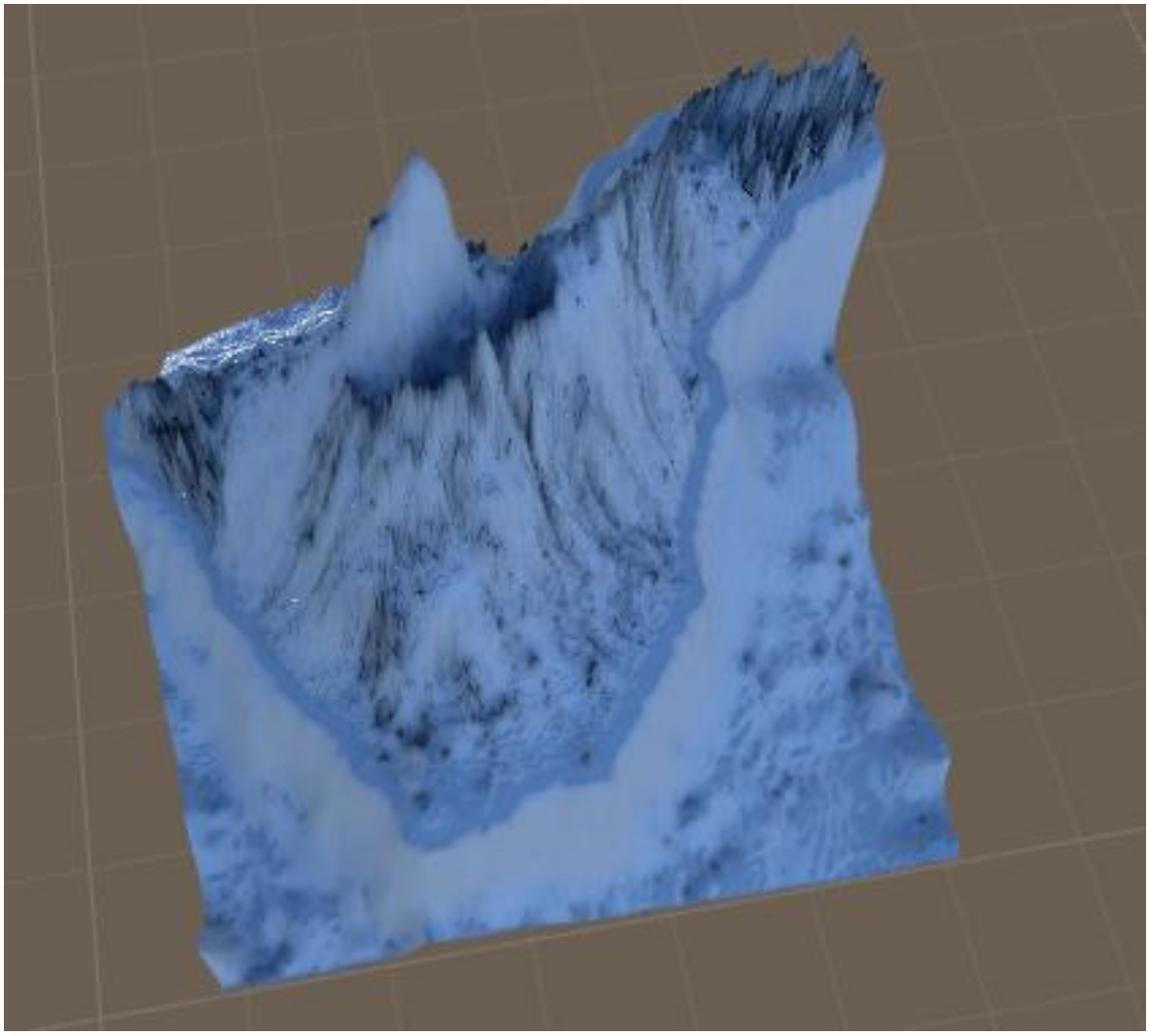

Figura 0-14: Terreno 3D creado a partir del mapa de altura de Tenerife en Unity.

Por ejemplo, nada más importarlo, Unity lo genera bastante abrupto, eso se puede solucionar disminuyendo el parámetro de Terrain Heigh, de nuevo en Inspector > Terrain Settings > Terrain Heigh.

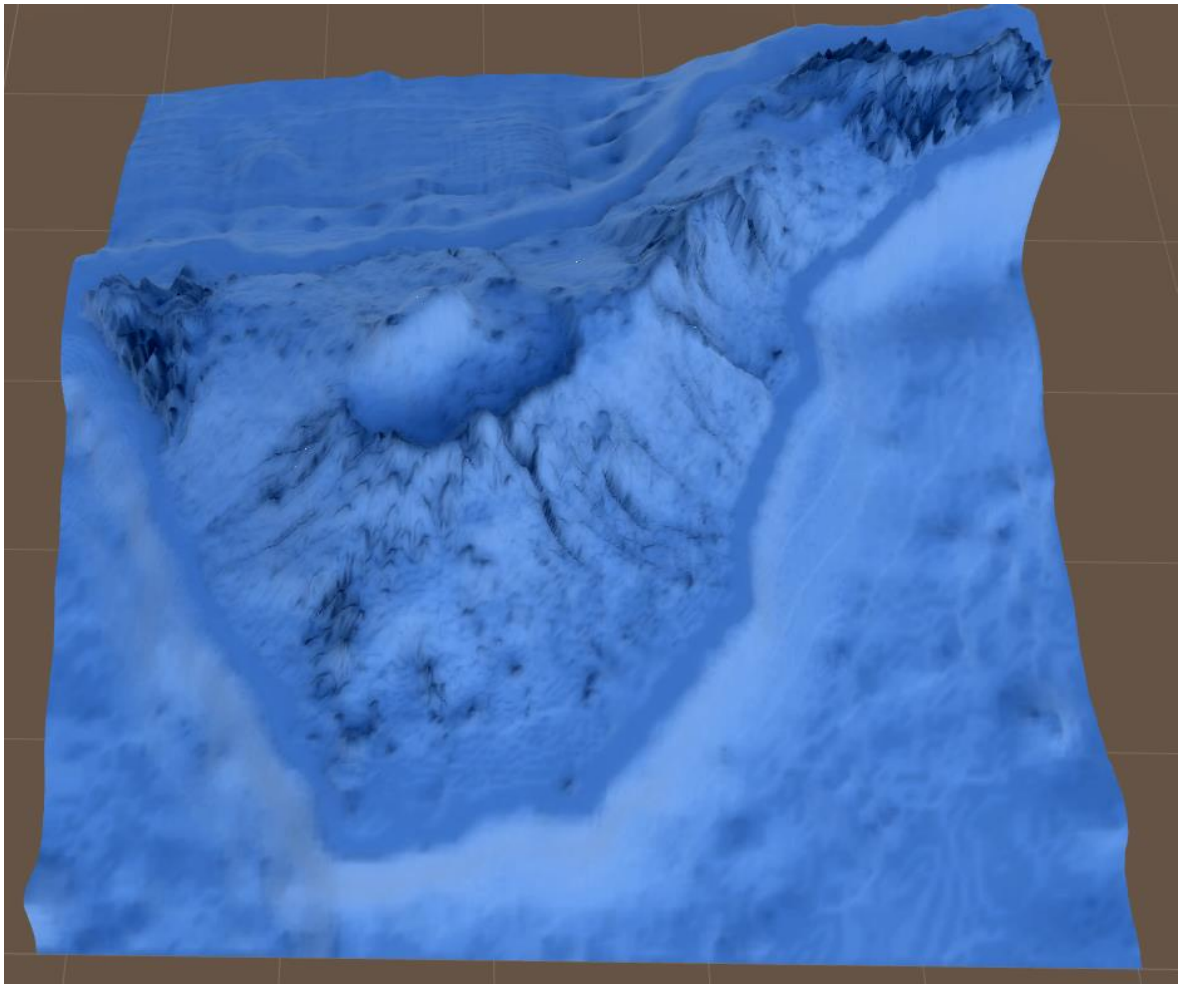

Figura 0-15: Terreno 3D de Tenerife suavizado.

Si le dedicamos algo más de tiempo y hacemos uso de las herramientas básicas de Unity, se puede obtener algo de este estilo.

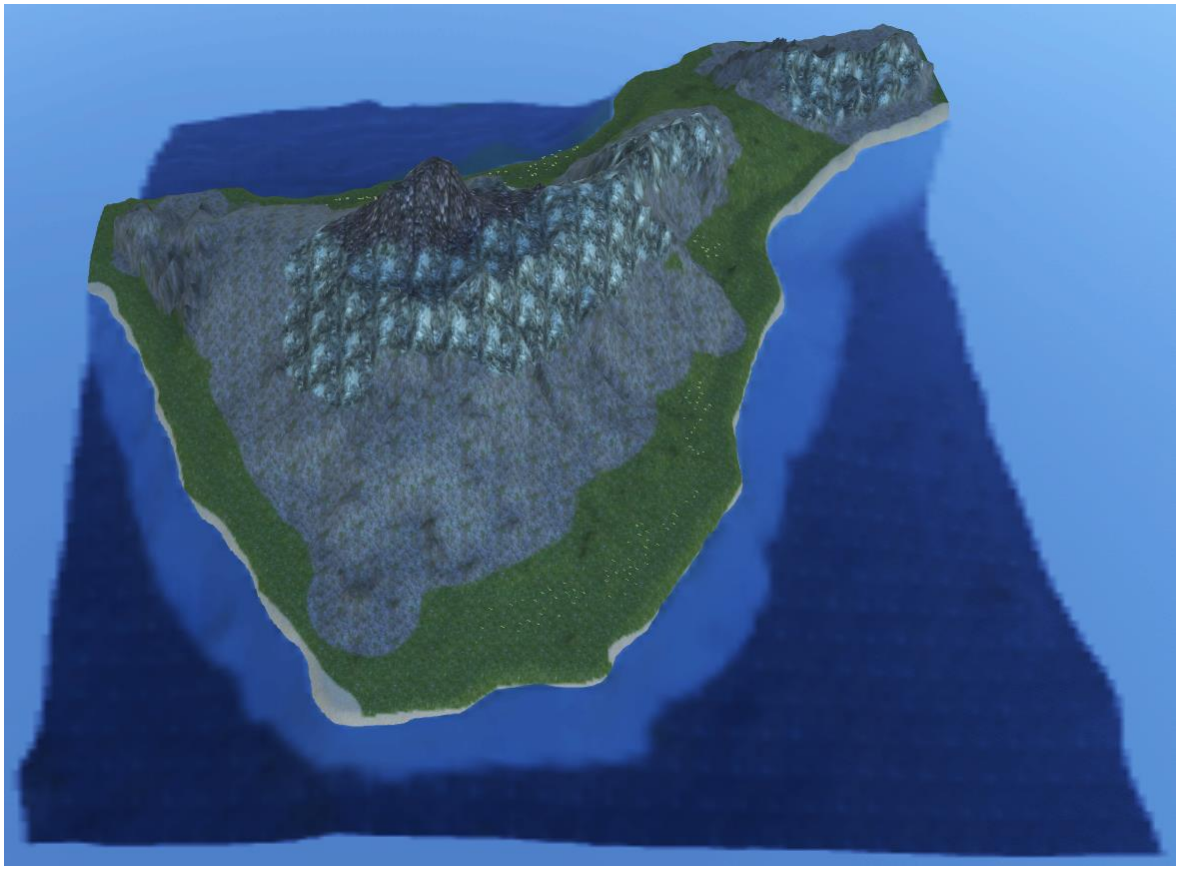

Figura 0-16: Ejemplo del modelo 3D generado de Tenerife adaptado con texturas y agua.